НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення веб-технологій та мобільних пристроїв»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Інструментальні засоби управління транспортними потоками»

Виконала:
студентка IV курсу, групи ТІ-61
Мікульська Марія Анатоліївна _____

Керівник:
доцент, кандидат економічних наук
Гусєва Ірина Ігорівна _____

Рецензент:
доцент, кандидат технічних наук
Веремійчук Юрій Андрійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки: 121 Інженерія програмного забезпечення

Спеціалізація: Програмне забезпечення веб-технологій та мобільних пристроїв

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020 р.

ЗАВДАННЯ

на дипломну роботу студенту

_____Мікульській Марії Анатоліївні_____
(прізвище, ім’я, по батькові)

1.Тема роботи Інструментальні засоби управління транспортними потоками
керівник роботи Гусєва Ірина Ігорівна, к.е.н., доц.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020 р. № **1168-с**

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи: WEB-система як засіб управління транспортними потоками

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) проаналізувати задачу розробки web-системи для обробки інформації про транспортні потоки та керування ними за допомогою розумних світлофорів.

5. Перелік ілюстративного матеріалу: актуальність теми дипломного проекту, мета роботи, поставлені задачі, GLOSA, схема архітектури web-сервісу, структура бази даних, діаграма прецедентів додатку, інтерфейс розробленого додатку, графік ефективності алгоритму, використані технології.

6. Дата видачі завдання ”11” жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	15.10.2019	
2.	Вивчення та аналіз задачі	16.10.2019 - 16.12.2019	
3.	Розробка архітектури та загальної структури системи	09.03.2020 - 23.03.2020	
4.	Розробка структур окремих підсистем	24.03.2020 - 10.04.2020	
5.	Програмна реалізація системи	14.04.2020 - 15.05.2020	
6.	Оформлення пояснювальної записки	18.05.2020 - 10.06.2020	
7.	Захист програмного продукту	17.05.2020	
8.	Передзахист	10.06.2020	
9.	Захист	15.06.2020	

Студентка _____ _Мікульська Марія Анатоліївна_____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _Гусєва Ірина Ігорівна_____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

В роботі розглянуто теоретичні та практичні аспекти розробки програмного продукту для управління транспортними потоками.

Метою дипломної роботи є розробка та впровадження web-сервісу для керування транспортними потоками, вхідними даними якого є метрики, що описують транспортну ситуацію, їх збереження та реалізація алгоритму із застосуванням GLOSA методології для передбачення точок критичного навантаження на відрізках доріг, оснащених регульованими світлофорами та подальшого динамічного керування цими транспортними потоками для оптимізації дорожньо-транспортної ситуації.

Пояснювальна записка складається зі вступу, п’яти розділів, висновків, списку використаних джерел, містить 54 сторінки, 14 рисунків, 7 таблиць та 4 додатки. При написанні звіту опрацьовано 12 джерел.

Ключові слова: транспортний потік, GLOSA, розумний світлофор, проміжний такт циклу роботи світлофорного об'єкта, RESTful API.

ABSTRACT

The report considers theoretical and practical aspects of software development to manage traffic flows.

The purpose of the thesis is to develop and implement a web-service for traffic flow management. The input data are metrics describing transport situation, their storage and usage for the algorithm based on GLOSA methodology to predict critical load traffic points on the road sections equipped with adjustable traffic lights.for further traffic flows management for traffic situation optimization cases.

This report consists of an introduction, five chapters, conclusions, a list of used sources and contains 54 pages, 14 illustrations, 7 tables and 4 appendices. 12 sources have been processed during writing the report.

Keywords: traffic flow, GLOSA, traffic light object intermediate cycle, smart traffic light, RESTful API.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

GLOSA	–	Green Light Optimal Speed Advisory
ЕГ	–	Екстрене гальмування
REST	–	Representational State Transfer
API	–	Application Programming Interface
UI	–	User interface
OAuth	–	Open Authorization

ВСТУП

В часи розквіту цифрової епохи кількість інтернет-користувачів неухильно зростає, а вартість інтернет-трафіку постійно зменшується. З огляду на цей процес абсолютна більшість членів суспільства, які мають доступ до інтернет-джерел тепер надають перевагу інформації з мережі, нехтуючи бібліотеками, телефонними довідниками, паперовими газетами, фізичними листами тощо. Ці умови є логічною підставою для появи все більшої кількості веб-сервісів з різними спрямуваннями та цільовою аудиторією, де навіть на перший погляд некомерційна тематика сайту може бути цікавою для бізнесу як з точки зору заробітку через підписки та платні функції реалізованого в браузері сервіса, так і через рекламне інтегрування.

Завдяки фреймворкам та скриптуванню ці сервіси не лише дозволяють перенести велику кількість задач із десктопних додатків, як от обробка мультимедіа і збереження файлів в онлайн, а й відкривають раніше недоступні можливості, наприклад, здійснення інтернет-платежів, пошук втраченого телефона або побудову оптимального маршруту на основі даних, отриманих з навколишнього середовища та від користувачів системи, яка і є темою на індивідуальне завдання дипломної роботи.

Транспортні затори — це комплексна проблема урбаністичного, економічного та екологічного характеру, що призводить до сталих перевантажень транзитних систем перевезень і зумовлена широким спектром чинників, найбільш поширеними серед яких прийнято вважати інженерні помилки при розробці та будівництві міської інфраструктури, збільшення населення мегаполісів та швидкі темпи розвитку економіки. Наслідками цих чинників стає надмірна кількість індивідуальних транспортних засобів, значно більша за пропускну спроможність міських доріг, обмеження руху транспорту через проведення вимушених

будівельних чи ремонтних робіт та автомобільні аварії, що часто блокують дорожні артерії в пікові години транспортного руху на невизначені терміни.

Програмний продукт, спроможний реагувати на зміни характеру та обсягів транспортних потоків, зібраних за допомогою мобільних пристроїв або інших обчислювальних пристроїв учасників дорожнього руху дозволить керувати світлофорами, скорочуючи або зменшуючи тривалість їх сигналів, надавати водіям прогнози щодо їх роботи, що може стати основою для системи сповіщень водіїв про затори і вибір більш оптимального шляху. Це дасть змогу більш рівномірно розподіляти транспортні потоки, економити час та пальне і навіть рятувати людські життя, уникаючи виникнення ДТП. Для отримання необхідних параметрів достатньо наявності мобільного пристрою та стабільного інтернет-з’єднання задля забезпечення безперебійного збору даних.

Розроблене та впроваджене програмне забезпечення для збору потрібних для роботи алгоритму керування транспортними потоками метрик, їх зберігання, обробка та підготовка результату, який буде максимально швидким, корректним та надійним може бути використане для корекції транспортних ситуацій на найбільш завантажених відрізках вже існуючих автомобільних доріг, а також слугувати предиктивним інструментом при побудові нових автомобільних магістралей, розв’язок тощо.

1. ПОСТАНОВКА ЗАДАЧІ УПРАВЛІННЯ ТРАНСПОРТНИМИ ПОТОКАМИ

Метою дипломної роботи є розробка web-системи для збору метрик, що характеризують транспортні потоки та їх подальшої обробки по алгоритму на основі GLOSA, який дозволяє передбачити пікові транспортні навантаження на відрізках доріг, оснащених світлофорами з можливістю приймати інтернет сигнал. Результати, отримані в процесі відпрацювання алгоритму, форматуються і передаються на світлофор, що дозволяє переналаштувати його графік роботи, таким чином згладжуючи піки транспортного навантаження.

Для досягнення мети дипломної роботи в процесі розробки було виконано такі задачі:

1. Отримання інформації про геолокацію користувача web-додатку, відображення на основі цих даних мітки на карті, позначки про місто та оцінку транспортних заторів в цьому місті по шкалі від 1 до 10.
2. Розробка форми для прокладання користувачем маршруту, де він може вказати пункти відправки та призначення відповідно.
3. Збір інформації про місце розташування і графік переключень світлових показників наявних в місті розумних світлофорів та внесення їх в базу даних системи.
4. Збір інформації про трафік на дорогах міста та відомі прокладені в його межах маршрути.
5. Розробка алгоритмів для пошуку розумних світлофорів на маршруті та передбачення завантаженості світлофора в момент часу.
6. Визначення світлофорів з піковим навантаженням та побудова нового розкладу на піковий проміжок часу.

7. Побудова маршруту, яким цікавиться користувач, проставлення відміток розумних світлофорів, які зустрічаються на цьому маршруті, видача інформації про довжину та середню тривалість поїздки, відображення старих та нових графіків світлофора, якщо вони потребували зміни.
8. Побудова результуючих математичних графіків з метою показати наочно, яких позитивних результатів по розумному керуванню транспортними потоками можна досягти, якщо використовувати розроблений в рамках дипломної роботи web-додаток.
9. Збір додаткових узагальнюючих метрик, таких як кількість запланованих поїздок, кількість користувачів що діляться інформацією про свої переміщення, число розумних світлофорів, до яких система має доступ і може змінити їм графік роботи.

З метою чіткого визначення цілей та завдання дипломного програмного продукту було складено технічне завдання, яке дозволило виробити стратегію розробки на основі вимог кінцевого користувача до системи, впорядкувати та доповнити їх, оцінити обсяг робіт, скласти план, визначитися з інструментальними засобами, які спроможні максимально задовольнити умовам та архітектурними рішенням, зробити висновки про терміни виконання.

Вхідні дані надходять до web-додатку з його інтерфейсу (адреси пунктів відправки та пунктів призначення), беруться з бази даних (інформація про розумні світлофори та їх графіки роботи, прокладені іншими пересічні маршрути, користувацькі дані) та отримуються зі сторонніх сервісів (місце розташування користувача, оптимальний маршрут, бал серйозності транспортних заторів за 10-бальною шкалою, географічні координати за адресою, модель трафіку).

Вихідні дані відображаються у вікні браузера (для користувача web-системи) і надаються у формі JSON-файлу (для подальшого використання при програмування світлофора).

З метою відповідності вимогам до надійності системи було передбачено авторизаційні механізми, функцію відновлення паролю до системи, контроль введеної інформації шляхом валідації, надання підказок при введенні, відсилання запиту на дозвіл використовувати геолокацію користувача, передбачено обмеження на потенційні некоректні дії під час роботи з системою.

Програмний продукт написано для використання через браузер і адаптовано інтерфейс під використання з мобільних пристроїв, а отже мінімальна конфігурація відповідає вимогам до користувацького браузера.

Ефективність програмного продукту визначається мірою зручності використання системи для виконання поставлених перед ним завдань у вимогах користувача.

2. МЕТОДИ ТА ЗАСОБИ УПРАВЛІННЯ ТРАНСПОРТНИМИ ПОТОКАМИ

2.1. Аналіз проблеми транспортних колапсів

Хоча безпосередньою причиною використання різноманітних дорожніх сигналів є необхідність у контролі конкуруючих транспортних потоків, паралельно такі міри дозволяють практикувати стиль водіння, при якому учасники дорожнього руху не вимушені здійснювати постійні зупинки на своєму шляху. Це дозволяє зменшити час у дорозі, адже зникає фактор як потреби у самих зупинках, так і постійного набору швидкості руху з нуля.

На додачу, такий підхід теоретично спроможний зменшити кількість “вузьких місць” та скоротити витрати пального, яке за інших, менш сприятливих умов, часто витрачається на простоях.

Нарівні з такими класичними засобами вирішення урбаністичних проблем цього спектру, як дорожні знаки швидкості руху та регулювальниками на найбільш завантажених вузлах міських автомобільних доріг, має сенс розглядати й більш інноваційний підхід, в методику якого покладено використання кишенькових пристроїв учасників дорожнього руху в особливих цілях.

Володіючи інформацією про те, через який проміжок часу найближчий світлофор загориться зеленим, водій зможе підібрати оптимальну швидкість руху, щоб в результаті досягнути відрізку дороги зі світлофором в момент, коли рух автомобілям дозволено, не будучи змушеним робити зупинку і перечікувати червоне світло. Альтернативний шлях використання отриманої з мобільних пристроїв

інформації — надання порад по вибору аналогічного маршруту, якщо ситуація зі світлофорами на найбільш оптимальному шляху наразі менш втішною.

Описана методологія носить назву GLOSA (Green Light Optimal Speed Advisory). Звісно, вона вже в певній мірі реалізована, і зробили це розробники програмного сервісу під назвою SignalGuru (рисунок 2.1).

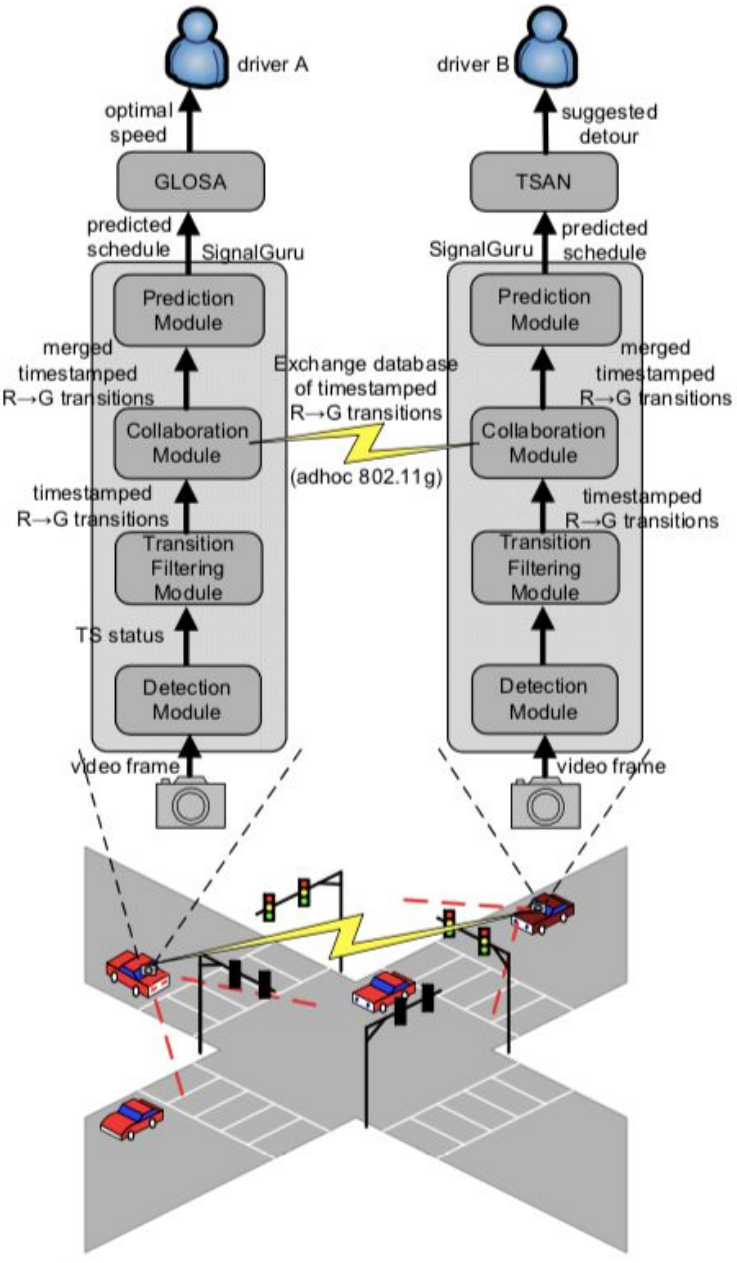

Рисунок 2.1 — Архітектура сервісу SignalGuru

SignalGuru передає інформацію, проаналізовану на основі різноманітних дорожніх сигналів на телефон водія або будь-які інші обчислювальні пристрої, що

знаходяться на борту транспортного засобу, наприклад автомобільний навігатор.

Сервіс працює за декількома різними алгоритмами паралельно. Ці алгоритми. в свою чергу, мають різні вимоги щодо точності прогнозування та максимального часу, на який можна отримати прогноз. Окрім GLOSA, SignalGuru володіє алгоритмом, котрий інформує водія про те, як багато поїздок в околицях заплановано кимось крім нього. Ще один алгоритм спроможний повідомити водієві, коли червоне світло припинить горіти, якщо він все таки був змушений здійснити зупинку на світлофорі. Завдяки ньому, водій може вимкнути двигун з метою економії двигун і розрахувати, коли пора заводити двигун заново і ще не настільки пізно, щоб стати причиною затору на відрізку дороги позаду нього.

Надаючи водіям інформацію про розклад, за яким світлофори змінюють колір світла, варто пам’ятати, що це може спокусити їх перевищувати швидкість у випадках, коли водій усвідомлює наперед, що при поточній швидкості руху він прибуде до світлофора, коли горіння зеленого світла вже закінчиться. Хоча цей недолік стосується в першу чергу самих алгоритмів, він, тим не менш, виступає і негативним фактором самого сервісу в цілому.

На основі проведеного аналізу аналогу, було створено технічне завдання із врахуванням всіх його переваг та недоліків. В якості міри для уникнення негативних наслідків, було розглянуто схему надсилання консультативних попереджень тим водіям, які наближаються до небезпечних показників швидкості руху.

2.2. Алгоритм розрахунку тривалості циклу світлофора

Основним завданням, яке було поставлено перед програмним забезпеченням дипломного проекту є досягнення підвищення рівня ефективності роботи світлофорів за рахунок оптимізації їхньої пропускної здатності на регульованих

розумними світлофорами відрізках дороги, що має на увазі раціональне та повне використання циклів експлуатації світлофорів.

Згідно до національних стандартів про встановлення дорожніх світлофорів (для України це ДСТУ 4092-2002), вони призначені для регулювання руху пішоходів та транспортних засобів на дорогах, вулицях, перехрестях, залізничних переїздах, відрізках дороги, де проїжджа частина звужується, в умовах поганої видимості тощо. Так, світлофори використовуються як інструмент регулювання на найбільш проблемних ділянках дороги, які є епіцентром виникнення найбільш складних та небезпечних дорожньо-транспортних пригод.

Як правило, обчислення, на основі яких будуються системи керування світлофорів не беруть до уваги таке урбаністичне явище, як години пік. З метою зробити систему графіків переключення світлових сигналів світлофорів більш гнучкою та динамічною, було розглянуто удосконалену методологію, яка враховує необхідність забезпечення можливості перетину відрізків доріг, оснащених світлофорами всіма транспортними засобами, які прямують з конфліктуючих напрямків в рамках одного циклу.

Застосування механізму уповільнення швидкості пересування транспортних потоків в правилах дорожнього руху відоме, як ЕГ.

Методологія, що буде розглянута в рамках дипломної роботи відрізняється від методології Ф. Вебстера, основними недоліками якої є похибка при застосуванні на дорогах з декількома односторонніми полосами та не врахування ступеня завантаженості альтернативних доріг і є її модифікацією.

Ефективність встановленого графіку роботи світлофора оцінюється можливістю його перетину всіма транспортними засобами, що стали в чергу на проїзд світлофора за один повний цикл переключення. Таким чином, саме динамічні розрахунки графків є основоположним інструментом для згладжування кривої пікових навантажень з метою уникнення транспортних інцидентів та неефективного використання ресурсів палива та часу, проведеного в заторах.

Згідно до діючих методичних вказівок тривалість повного циклу світлофора не повинна перевищувати 120 секунд. Ґрунтуючись на відомостях про активність транспортних засобів на протязі години в найбільш завантаженому альтернативному маршруті N_a, перш за все обчислюється кількість учасників дорожнього руху що прибули до світлофора за визначений час:

$$N_a^i = \frac{N_a \cdot 120}{3600} \tag{2.1}$$

де N_a^i — це значення об'єму транспортного потоку за один цикл тривалістю в 120 секунд.

Виходячи з умови, що за один повний цикл транспортні засоби, які стали в чергу повинні перетнути світлофор в межах наступного циклу, його максимальна тривалість залежить від останнього автомобіля в черзі. Нехай перший автомобіль у черзі здійснив зупинку на лінії стоп перед світлофором. Тоді місце розташування останнього транспортного засобу в черзі обчислимо з використанням наступної формули:

$$S_{enter}^i = n_i \cdot (B + l) \tag{2.2}$$

де n_i — кількість учасників дорожнього руху, які стали в колону перед світлофором за час одного циклу переключень;

B — середня довжина транспорту (в якості константи можна прийняти 4.5 м);

l — середня відстань між транспортними засобами, які стоять один за одним перед світлофором через обмежуючий рух сигнал (за константу можна взяти значення 1 м).

$$n_i = \frac{N_a^i}{\text{c}} \tag{2.3}$$

де N_a^i — це значення об'єму транспортного потоку в напрямку, що досліджується за період одного повного циклу;

с — кількість полос руху на дорозі в заданому напрямку.

Враховуючи коефіцієнт прискорення та затримку на відновлення попередньої швидкості руху після повної зупинки, автомобіль, який стоїть в голові очікуючої колони на смузі i встигає досягти того місця, перебуваючи на якому водій має право проїжджати світлофор за t_{oi} секунд, яких цілком достатньо, щоб вписатися в тривалість циклу переключень.

$$t_{oi} = \sqrt{\frac{2 \cdot (S_{enter}^i - S_{wait})}{a}} + t_{delay}^i \tag{2.4}$$

де $(S_{enter}^i - S_{wait})$ — відстань, пройдена останнім на смузі автомобілем з моменту відновлення руху після отримання дозволяючого сигналу світлофора і до досягнення лінії, після якої у водія фізично відсутня можливість зупинити свій транспортний засіб до моменту перетину місця, передбаченого для цього правилами дорожнього руху (лінія стоп, пішохідний перехід тощо), м;

a — прискорення транспортного засобу, м/ с^2 (для практичних розрахунків береться 1.5 м/ с^2);

t_{delay}^i — затримка для відновлення руху останнім транспортним засобом, с.

$$t_{delay}^i = n_i \cdot t_{delay}^{av} \tag{2.5}$$

де t_{delay}^{av} — середній час затримки, необхідний кожному автомобілю на смузі для відновлення руху, с (при практичних розрахунках можна взяти 1.0 с).

S_{wait} — відстань між автомобілем та рубежем, при досягненні якого згідно до правил дорожнього руху водій фізично неспроможний здійснити зупинку, а отже отримує право продовжувати рух, м:

$$S_{wait} = (t_1 + t_2 + 0,5t_{SLy}) \cdot \frac{V_a}{3,6} + \frac{V_a^2}{26 \cdot J_y} \quad (2.6)$$

де V_a — найбільш імовірна швидкість останнього в колоні транспортного засобу, якої він досягає в момент появи обмежувального сигналу на світлофорі, км/год (при розрахунках на практиці прийнято значення 40 км/год);

J_y — сповільнення транспорту при появі обмежувального сигналу, м/с^2 (при практичних розрахунках для сухого асфальтного полотна становить 4.6 м/с^2);

t_1 — час, за який водій відреагує на зміну ситуації (за замовчуванням 0.6 с);

t_2 — затримка відповіді двигуна автомобіля при початку руху, с (у розрахунках на практиці беруть 0.1 с);

t_{SLy} — час від початку сповільнення руху до повної зупинки, с (при розрахунках можна взяти значення 0.35 с).

Методологія дозволяє розрахувати час, необхідний для згладжування пікових навантажень на ділянках дороги, оснащених розумним світлофором із врахуванням таких параметрів, як конфігурація транспорту і кількість смуг на дорозі.

2.3. Алгоритм переналаштування світлофора на гнучкий графік

Корекція графіку роботи світлофору може виконуватися із врахуванням як мінімального, так і максимального часу, потрібного для переключення на новий

режим, при цьому мінімум становить не менше 7 с для будь-якої фази. При обчисленні максимального значення корекція здійснюється після оцінки можливості перетину дороги пішоходами, які починають свій рух при появі дозволяючого сигналу. Необхідно або збільшувати тривалість циклу переключення до потрібного значення, або змінювати модель руху пішоходів шляхом удосконалення відрізку дороги, додавши острівок пішохода, на якому люди матимуть змогу здійснити зупинку після перетину певної кількості смуг (ця альтернатива не розглядається в рамках дипломного проекту).

Для підрахунку часу, необхідного пішоходам для перетину дороги застосовано наступну формулу:

$$t_{ped} = 5 + \frac{S_{ped}}{V_{ped}} \tag{2.7}$$

де S_{ped} — відстань, пройдена пішоходом по дорозі під час зеленого світла світлофора, м;

V_{ped} — середня швидкість пішоходів при переході дороги на зелене світло, м/с (при розрахунках можна використовувати значення 1.3 м/с).

Загальна тривалість усіх основних циклів переключення дорівнює сумі тривалості циклів на альтернативних маршрутах:

$$\Sigma\, T_{mi} = t_{m1} + t_{m2} + ... + t_{mi} \tag{2.8}$$

де $t_{m1,}\ t_{m2,}\ t_{mi}$ — тривалість циклів по всім напрямкам на різних фазах.

Для спрощення розрахунку тривалості циклу на інших, менш завантажених напрямках враховується пропорційність транспортних потоків. Цей параметр визначається так званим одноразовим коефіцієнтом, фізичне значення якого є

відношенням максимального об'єму найбільш завантаженого транспортного потоку на одній смузі дороги до максимального об'єму аналогічної метрики у зворотному напрямку.

$$Y_i = \frac{N_{ai} \cdot c_1}{N_{a1} \cdot c_i} \tag{2.9}$$

де N_{a1} — об'єм транспортного потоку в найбільш завантаженим у напрямку за час одного циклу;

c_1 — кількість смуг дороги в найзавантаженішому напрямку;

N_{ai} — об'єм транспортного потоку в одному з досліджуваних альтернативних напрямків за один цикл;

c_i — кількість смуг в напрямку, який досліджується.

$$t_{mi} = Y_i \cdot t_{m1} \tag{2.10}$$

Цієї методики достатньо для зменшення максимально можливої тривалості циклу в 120 секунд на більш динамічну та обґрунтовану.

2.4. Висновки до розділу

В даному розділі було визначено предметну область дипломної роботи. В ході її детального аналізу було розглянуто класичні підходи до вирішення таких урбаністичних проблем, як надмірне завантаження міських автомобільних доріг,

вузькі місця на транспортних розв'язках, недоцільні витрати пального, підвищений ризик виникнення дорожньо-транспортних пригод тощо.

Було окреслено вектори розвитку більш інноваційних підходів, одним з яких є й використання інформації, наданої кишеньковими пристроями учасників дорожнього руху з метою побудови моделей трафіку, більш наближених до реальної ситуації на дорогах.

В якості теоретичної бази для застосування таких моделей на практиці було розглянуто GLOSA методологію і здійснено пошук та аналіз програмних продуктів на ринку, які вже реалізували її в ядрах своєї системи, найбільш вдалим та яскравішим представником з-поміж яких виявився сервіс SignalGuru. В розділі було розглянуто і альтернативні методики, використані цим сервісом, визначено його переваги та недоліки, які були враховані на етапі створення технічного завдання для дипломної роботи.

У підрозділі опису алгоритма розрахунку тривалості циклу світлофора було окреслено проблематику у використанні звичайних нерегульованих світлофорів в найпроблемніших зонах і описано методологію, яка є вдосконаленням методу Ф. Вебстера. За мету було поставлено скорочення стандартної тривалості повного циклу світлофора (120 секунд) до оптимального часу, розрахунки якого базуються на основі таких параметрів, як кількість смуг на дорозі та ступінь її завантаженості відносно альтернативних шляхів.

Оцінкою ефективності цієї методики є показник можливості проїзду ділянки дороги транспортним засобом в межах одного циклу світлофора, який став у колону очікування на попередньому циклі.

Підрозділ про алгоритм переналаштування світлофора на гнучкий графік описує спосіб ефективного балансування в чергуванні зеленого та червоного світла із врахуванням потреб як водіїв, так і пішоходів.

Здобута інформація утворила математичне ядро програмного продукту, розробленого в рамках дипломного проекту.

3. ПРОГРАМНІ ЗАСОБИ РОЗРОБКИ СИСТЕМИ

3.1. Інструменти серверної розробки

Під поняттям “серверна частина” проекту зазвичай мають на увазі операції, що виконуються сервером у взаємовідносинах клієнт-сервер у комп'ютерній мережі. Часом так називають кросплатформне середовище, що дозволяє користувачам взаємодіяти між собою. Серверна частина проекту включає в себе побудову функціональних потоків сервера (його архітектуру), баз даних та адміністративну панель для зручного керування внутрішніми процесами. Таким чином, серверна частина обов’язково включає в себе функції, що маніпулюють даними.

В якості архітектурного шаблону для програмного продукту дипломної роботи виступає MVC (рисунок 3.1).

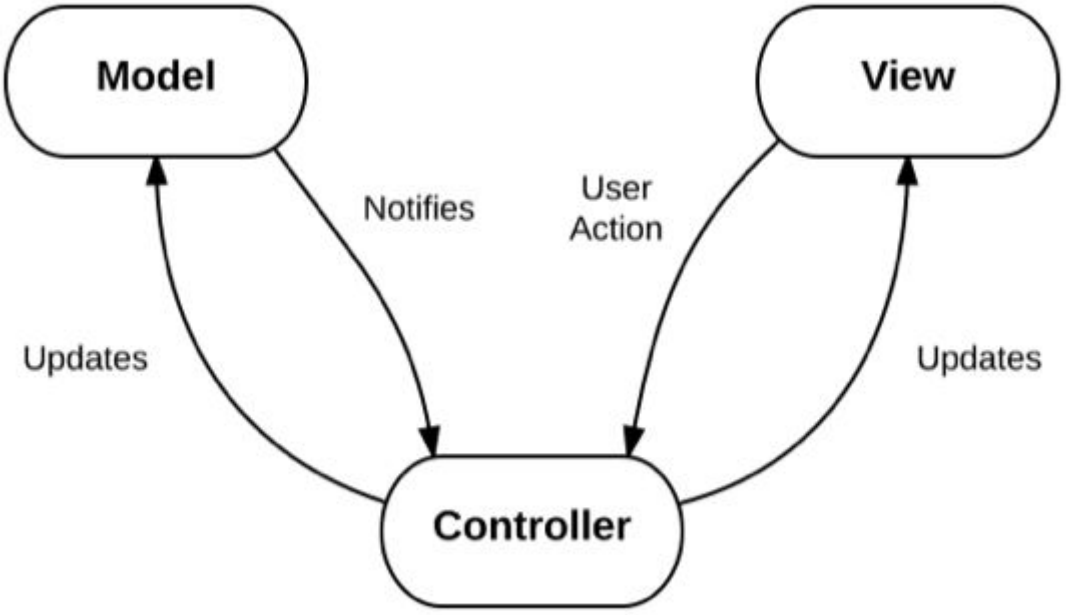

Рисунок 3.1 — Схема MVC паттерна

Такий вибір продиктований зокрема тим, що в стек серверних інструментів для розробки з-поміж інших входить і Laravel фреймворк, який підтримує дану парадигму в повній мірі. Однак найвагомішими аргументами на користь такого рішення все ж залишаються відчутний приріст в ефективності та швидкості розробки, в тому числі паралельної, спрощений процес налагодження та подальших оновлень додатку, можливість використовувати одні й ті самі компоненти для управління під будь-яким інтерфейсом завдяки відсутності форматування даних у відповіді на запит (в рамках дипломного проекту це front-end інтерфейс та прикладний програмний інтерфейс (API) одночасно).

Laravel — це безкоштовний PHP фреймворк з відкритим вихідним кодом, призначений для розробки веб-додатків за архітектурним шаблоном MVC, що передбачає поділ даних програми, призначеної для користувача інтерфейсу і керуючої логіки на три окремих компоненти: Модель, Представлення і Контролер. В якості основи Laravel виступають компоненти іншого фреймворка — Symfony.

Найбільш помітними та привабливими можливостями Laravel виступають:

1. Реалізація шаблону Eloquent ORM, яка дозволяє зручно маніпулювати об'єктами-сутностями бази даних;
2. Механізм автозавантаження класів дозволяє не підключати вручну файли через include і запобігає завантаженню компонентів, які по факту не викликаються в процесі конкретного сценарію роботи програми;
3. Зручна система міграцій допомагає спростити розгортання і оновлення веб-додатків, робити відкати у випадку виникнення помилок, тим самим реалізуючи своєрідну систему контролю версій для бази даних;
4. Наявність вбудованої підтримки движка шаблонів Blade, за допомогою якого можна робити прості веб-сторінки використовуючи спеціальний синтаксис.

Навколо фреймворка створена потужна екосистема, розробники створили для нього зрозумілу та вичерпну документацію.

В Laravel багато синтаксичного цукру. Тут немає довгих та складних конструкцій, лише короткі та продумані назви функцій. Фреймворк містить зручний механізм обробки помилок і виключень, включає в себе вбудовані механізми аутентифікації і авторизації користувачів, яку можна переналаштувати під свої потреби. З коробки він надає механізми для кешування через Memcached та Redis, надає зручні функції для використання простого файлового кешування даних.

Як правило, в якості моделей виступають інтерфейси збереження даних. В розробленому дипломному проекті таким інтерфейсом виступає база даних SQL. Моделі реалізовано у вигляді класів, кожний з яких репрезентує таблицю БД або сукупність таблиць, а властивості класу відповідають колонкам цих таблиць. Таблиці мають доволі тісні зв’язки одна з одною, що дозволяє забезпечити цілісність даних. Laravel-компонент під назвою Eloquent спрощує встановлення та керування цими зв’язками. Розробка бази даних велася через міграції, беззаперечний плюс яких полягає в тому, що крім створення безпосередньо самих таблиць, існує також можливість керувати версіями бази даних.

Контролери відповідають за керування бізнес-логікою програми та реагування на дії користувачів системи. Кожний публічний метод контроллера є точкою входу для запитів через заздалегідь прописані Routes. Параметри в тілі цих запитів, як правило, відповідають аргументам метода. Так, при потребі відмалювати в браузері профіль користувача, необхідно надіслати запит на route, підв’язаний до відповідального за цю дію контролера та його метода. Саме контролер, перехопивши управління, ініціює запит до бази даних, відправить отримані дані на обробку сервісом статистики і у якості відповіді поверне відформатований набір даних, записи в якому потім достатньо просто розташувати на потрібні місця в інтерфейсі.

Простими словами, View (представлення) — це інтерфейс користувача, на якому клієнт (якщо мова іде про API) або користувач (якщо мається на увазі front-end інтерфейс) може виконувати деякі дії. Єдине завдання, яке покладають на

представлення — відображення даних. Для реалізації представлень Laravel постачає шаблонізатор Blade. Його файли HTML-подібні, але містять і PHP-подібні синтаксичні конструкції (if, foreach, switch case), за допомогою яких отримана з контролера відповідь відмальовується в інтерфейсі. Крім цих трьох основних елементів, архітектура передбачає декілька допоміжних сутностей, які, тим не менш, продовжують ідеологію парадигми MVC і виступають її надбудовами: Middlewares — зручний механізм для фільтрації HTTP-запитів додатку, наприклад з метою перевірки на авторизованість; Services — класи, які описують безпосередньо самі алгоритми обробки даних; Providers — PHP обгортки для HTTP клієнтів, які є посередниками між web-сервісом та API сторонніх web-сервісів тощо.

За допомогою менеджера пакетів для PHP під назвою Composer, фреймворк Laravel дозволяє легко встановлювати і підключати сторонні пакети з packagist.org, будь-якого git, mercurial або svn сховища. Оскільки всі пакети встановлюються в поточну директорію (де було виконано команду install), програміст отримує можливість використовувати кілька різних версій бібліотек при роботі над різними проектами паралельно. За потреби команда update оновлює всі встановлені (або встановить випадково видалені) пакети до свіжих версій (рисунок 3.2).

Рисунок 3.2 — Схема роботи пакетного менеджера Composer

Саме через Composer було встановлено laravel/framework пакет та Guzzle — PHP HTTP клієнт, який спрощує надсилання HTTP-запитів та тривіальну задачу інтеграції з веб-службами. Простий інтерфейс для створення рядків запитів, POST-запитів, потокового завантаження, завантаження великих тіл запиту, використання файлів cookie HTTP, завантаження даних JSON тощо. Guzzle може надсилати як синхронні, так і асинхронні запити, використовуючи один інтерфейс. Guzzle абстрагує базовий протокол HTTP, за рахунок цього відсутня жорстка залежність від потоків CURL, PHP та сокетів.

Часи, коли весь код web-сервісу зберігався в єдиному файлі давно минули. Нарівні із впровадженням нових парадигм програмування, розробкою конвенцій про чистоту коду та прийняттям використання систем керування версіями за корпоративний стандарт стрімко розвивалася й інфраструктурна культура, покликана спрощувати найбільш непередбачувані, важкодоступні та складні елементи в налаштуванні програмної екосистеми.

Docker — це програмне забезпечення з відкритим вихідним кодом, яке дозволяє розробляти, доставляти, тестувати та запускати додатки, слідуючи парадигмі контейнеризації, тим самим назавжди вирішуючи проблему залежності програмного продукту від середовища, в якому він був запущений (рисунок 3.3). Docker є більш оптимальною альтернативою для віртуальних машин. На відміну від апаратної віртуалізації, він запускає процеси на рівні основної операційної системи, а отже уникає таких проблем віртуалізації, як тривалий час завантаження, неефективне використання ресурсів та занадто складний процес налаштування.

Контейнер — це стандартна одиниця програмного забезпечення, що пакує код та всі його залежності в один, легко відтворюваний в інших системах, об’єкт. Будучи спроектованою в одному обчислювальному середовищі, кінцева програма працюватиме так же швидко та надійно у будь-якому іншому, де буде піднято запрограмований контейнер.

Зображення контейнера Docker — це легкий, автономний, виконуваний пакет програмного забезпечення, який включає все необхідне для запуску програми: код, час виконання, системні інструменти, системні бібліотеки та налаштування.

Зображення контейнерів стають контейнерами під час виконання, а у випадку Docker — коли запускаються на демонах Docker Engine.

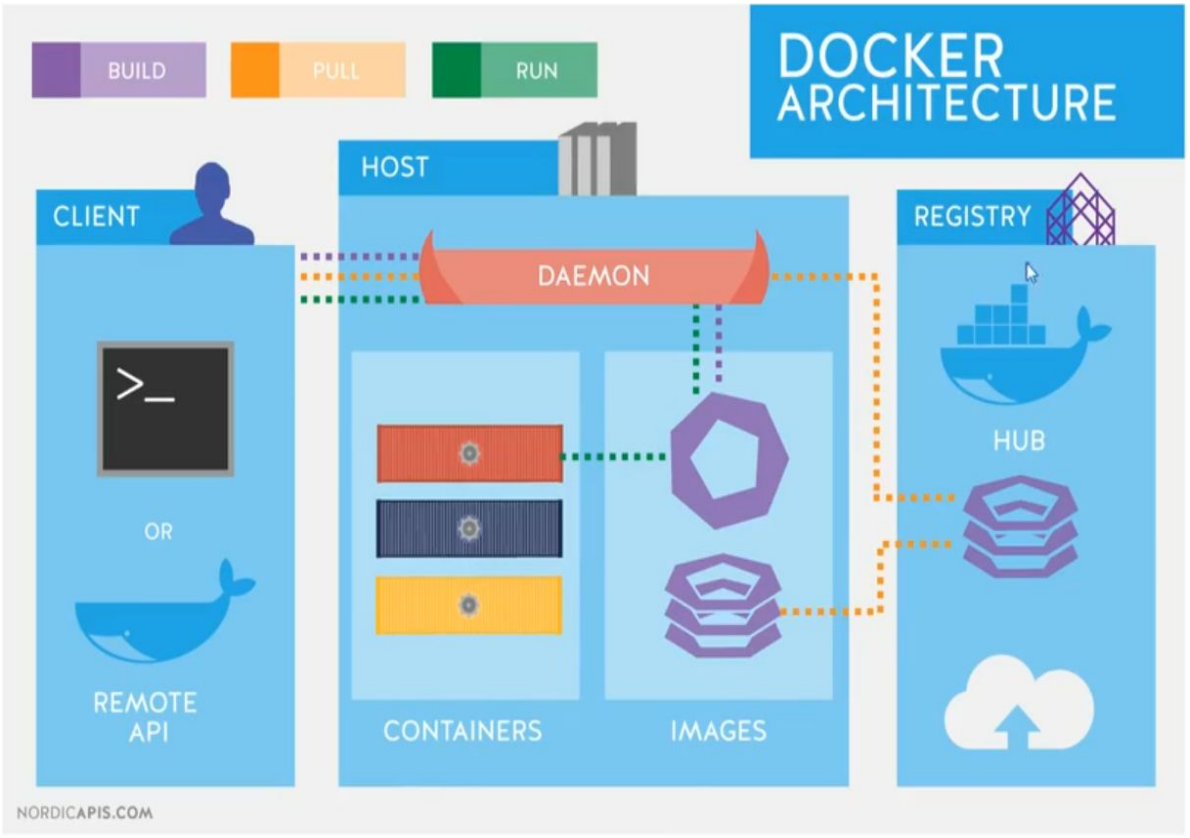

Рисунок 3.3 — Архітектура класичного докеризованого додатку

Таким чином, для забезпечення контейнерної архітектури достатньо налаштувати docker-compose.yml файл та створити контейнер на основі офіційного image, який можна вільно обрати на docker hub. Оскільки в якості мови програмування для реалізації API було обрано php, за основу для контейнера додатка дипломного проекту використовуватиметься webdewops/phpnginx:7.4. Інтеграція з MySQL-подібною реляційною базою даних та СКБД phpMyAdmin організована аналогічним способом.

Для розробки UI елементів системи, пов'язаних з картами, маршрутами, геолокацією та іншими транспортними даними частково використовувалися API сторонніх сервісів.

Так, канва мапи для проставлення відміток геолокації та прокладання маршруту з його наступним відображенням була відмальована засобами бібліотеки Google Maps Javascript API.

Інформацію про відстань від точки відправки до точки прибуття та час, який в середньому займає цей маршрут при відсутності серйозних навантажень на транспортну систему було отримано через RESTful API сервісу TomTom Navigation.

Процес прокладання маршруту здійснено на основі даних, отриманих одразу від двох систем — backend бібліотеки Google Directions API та Routing API TomTom Navigation сервісу, серед результатів яких потім обирався найбільш оптимальний. Варто зауважити, що результуюча вибірка при прокладанні маршруту враховувала також інформацію про транспортні затори, яку сервіс Google отримує в режимі реального часу, а сервіс TomTom — беручи до уваги історичну інформацію про затори на цьому відрізку дороги в тому числі. Оскільки прокладений шлях транспортні API, як правило, подають у вигляді послідовності гео-точок, між якими маршрут прокладено по прямій (простий напрямлений граф, вершинами якого є географічні координати), необхідно вирішити задачу переведення адреси, поданої користувачем у вигляді рядка через input елемент у формі прокладання маршруту в координати на мапі. Вирішити цю проблему допомогло використання обширної бази даних з адресами, підказки по ним при введені користувачем маршруту, який його цікавить та конвертер геолокації в адресу і навпаки, який написано з використанням TomTom Navigation Search API.

Оскільки жоден транспортний API, в тому числі з-поміж платних не володіє інформацією (або не надає її у відкритому доступі) про місце розташування на мапі світлофорів, в рамках розробки web-сервісу для дипломної роботи було самостійно

проведено збір цих даних з використанням Google Maps у режимі перегляду вулиць і збережено в базу даних системи для управління транспортними потоками вручну.

3.2. Інструменти для розробки користувацького інтерфейсу

HTML застосовується для створення веб-сторінок і логічного розбиття сайту на структурні елементи та інтерпретується браузером для відображення у вигляді документа в зручній для людини формі. Інструктуюча одиниця HTML —теги.

CSS — це мова опису зовнішнього вигляду документа, написаного мовою розмітки. Основою CSS виступають каскадні таблиці стилів, покликані відокремити опис логічної структури веб-сторінки від її зовнішнього вигляду. Кожне стильове правило задається парою “властивість — значення”, а ці правила в подальшому застосовуються до селекторів.

У всі основні браузери вбудовані інтерпретатори Javascript, але насправді це повноцінна мова, яка може використовуватись в тому числі і за їх межами. Javascript також іноді називають безпечною мовою програмування, оскільки вона не надає низькорівневих засобів роботи з пам’яттю, процесором, адже є орієнтованою в першу чергу на браузери, де в подібних функціях відсутня гостра необхідність. Засобами мови скриптів можна реалізувати створення нових HTML-тегів та видалення існуючих, редагування стилів елементів, їх приховування, розпізнавання різноманітних дій користувача на сторінці та відповідна реакція на них (кліки миші, переміщення курсора, натискання на клавіатуру), посилання запитів на сервер і підтримка актуалізації даних без перезавантаження сторінки (ця технологія називається "AJAX"), виведення повідомлень тощо.

В свою чергу JQuery — це багатофункціональна бібліотека JavaScript, яка завдяки простому у користуванні API дозволяє здійснювати маніпуляції, перераховані вище значно простіше.

3.3. Висновки до розділу

В розділі описано процес вибору інструментів програмування для створення web-сервісу та переваги обраних засобів над альтернативними.

За архітектурний шаблон було обрано MVC, оскільки він в повній мірі підтримується backend фреймворком Laravel, дозволяє вести процес розробки паралельно та органічним шляхом вичленити концептуальні блоки, розділені як за формою, так і за призначенням. Моделі було реалізовано у вигляді класів, які репрезентують сутності бази даних JSON-файли, тіло яких отримується зі сторонніх транспортних API сервісів. Контролери, які відповідають за керування бізнес-логікою було описано таким чином, щоб кожний метод відповідав певному endpoint розробленої системи, а представленнями послужили форми з UI елементами, які сформували кінцевий інструмент для взаємодії користувача та web-додатку. З метою організації зручного використання сторонніх програмних ресурсів було впроваджено менеджер пакетів Composer, а за його допомогою з-поміж інших встановлено і PHP HTTP клієнт Guzzle. В якості менеджера сервісної інфраструктури вибір було зроблено на користь Docker, який підтримує парадигму контейнеризації, розв'язуючи дилему залежності створеного програмного забезпечення від апаратного, на якому воно було написане. Стек Front End технологій крім класичних HTML і CSS було доповнено мовою JavaScript та бібліотекою JQuery.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ УПРАВЛІННЯ ТРАНСПОРТНИМИ ПОТОКАМИ

4.1. Архітектура системи

Основна ідея web-сервісу полягає в тому, щоб на основі інформації про дорожньо-транспортну ситуацію сформувати сигнал для розумного світлофора про зміну його графіку роботи в час виникнення пікового навантаження в околицях його місця розташування (рисунок 4.1).

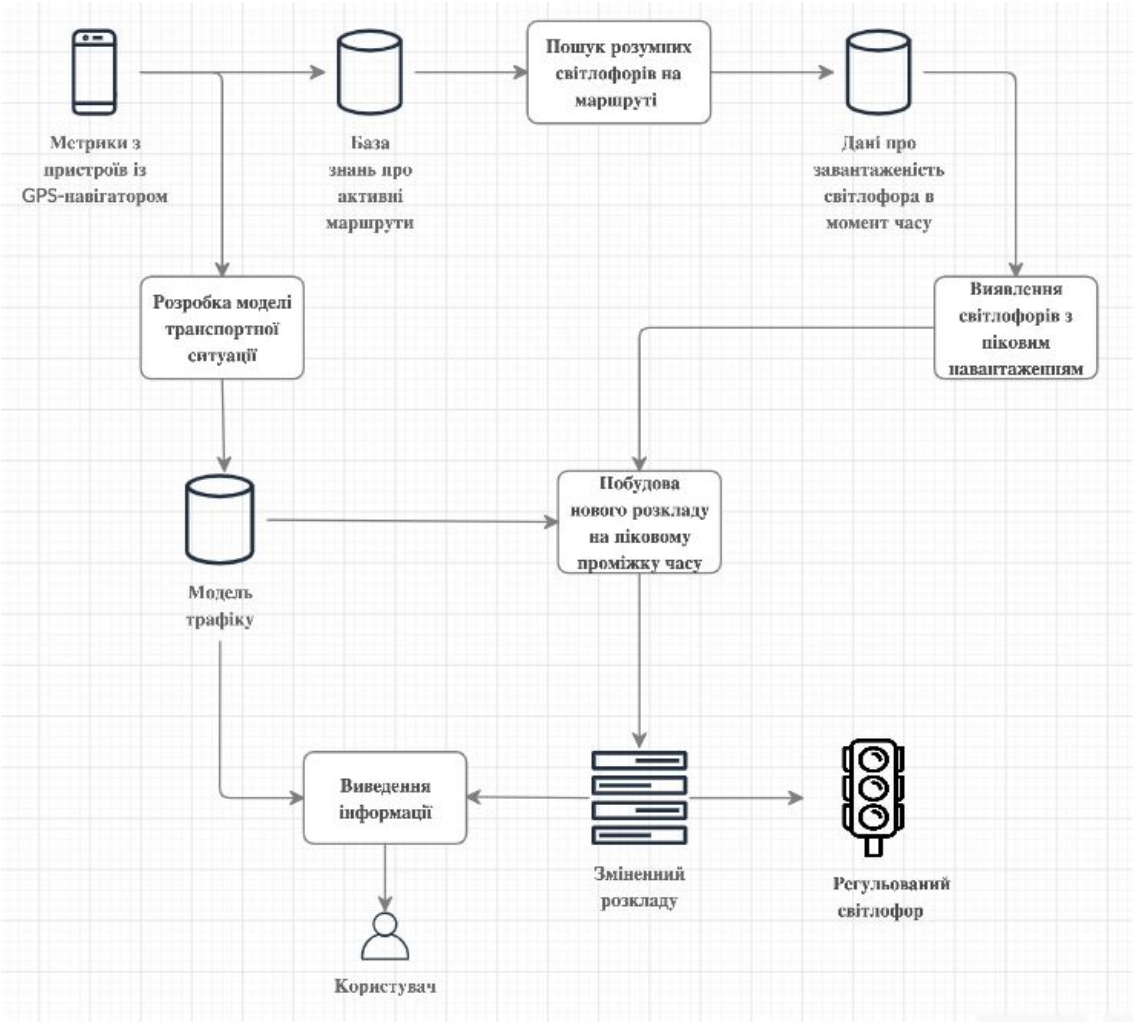

Рисунок 4.1 — Архітектура web-сервісу

Відправною точкою для транспортних алгоритмів виступають метрики з мобільних пристроїв, які знаходяться на борту транспортних засобів і є пасивними учасниками дорожнього руху, а отже володіють метриками, що описують місце розташування у просторі в певний момент часу, задають траєкторію руху і можуть повідомити про швидкість пересування.

Ця інформація використовується для побудови двох моделей — моделі транспортної ситуації та бази знань про маршрути, рух за якими здійснюється в певний проміжок часу.

Остання модель в свою чергу виступає в ролі вхідної інформації для алгоритму пошуку регульованих світлофорів, які лежать на запланованих маршрутах. Вихідними даними цього алгоритму є рівень навантаженості розумного світлофора в момент часу, котрі потім фільтруються, залишаючи лише ті одиниці, на яких спрогнозовано пікове навантаження. В сумі з моделлю трафіку, ці дані використовуються в алгоритмі, який розраховує новий розклад для світлофора таким чином, щоб згладити навантаження на транспортну систему в точках їх розташування в пікові відрізки часу.

Заново сформований розклад переключень світлофора форматується для передачі мережевого сигналу безпосередньо на сам регульований світлофор, а також відображається користувачу системи поруч із даними про трафік.

4.2. Алгоритм роботи програми

Алгоритм роботи програми ґрунтується на логіці, закладеній в архітектуру web-системи і формально описаний в блок-схемі алгоритму (рисунок 4.2).

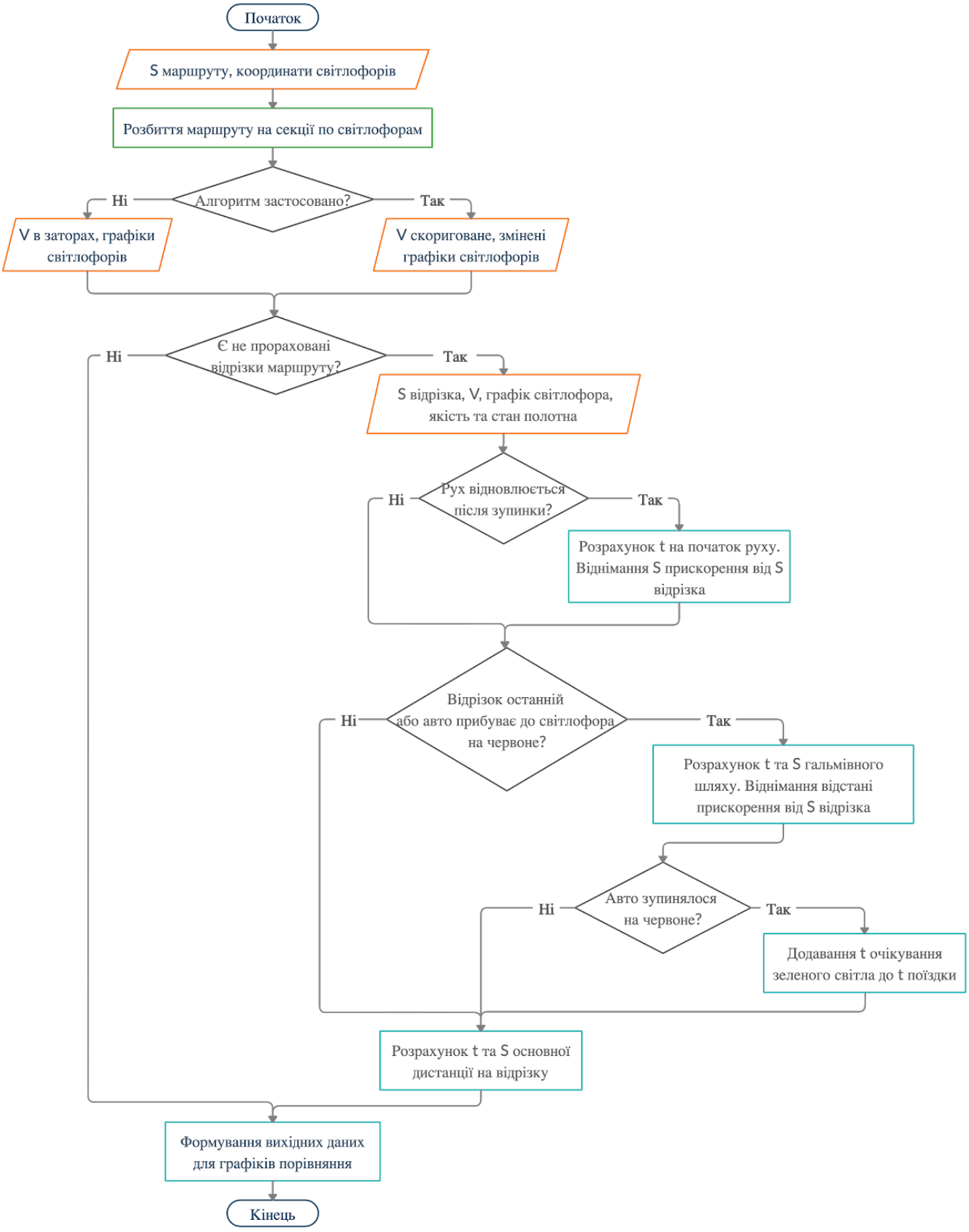

Рисунок 4.2 — Алгоритм роботи програми

4.3. Структура бази даних

Фізичною моделлю для дипломного проекту виступає транспортна екосистема (рисунок 4.3). Оскільки у web-сервісі реалізовано можливість здійснювати авторизацію у систему, база даних містить також декілька системних таблиць, в яких зберігаються дані користувача, потрібні для входу та відновлення доступу до сайту та деяка кількість персональної інформації, на основі якої створюється його профіль. Тим не менш, основні таблиці бази даних репрезентують моделі реального світу і відображають в собі метрики, потрібні для розрахунків алгоритму перепрограмування світлофорів з метою розумно керувати транспортними потоками.

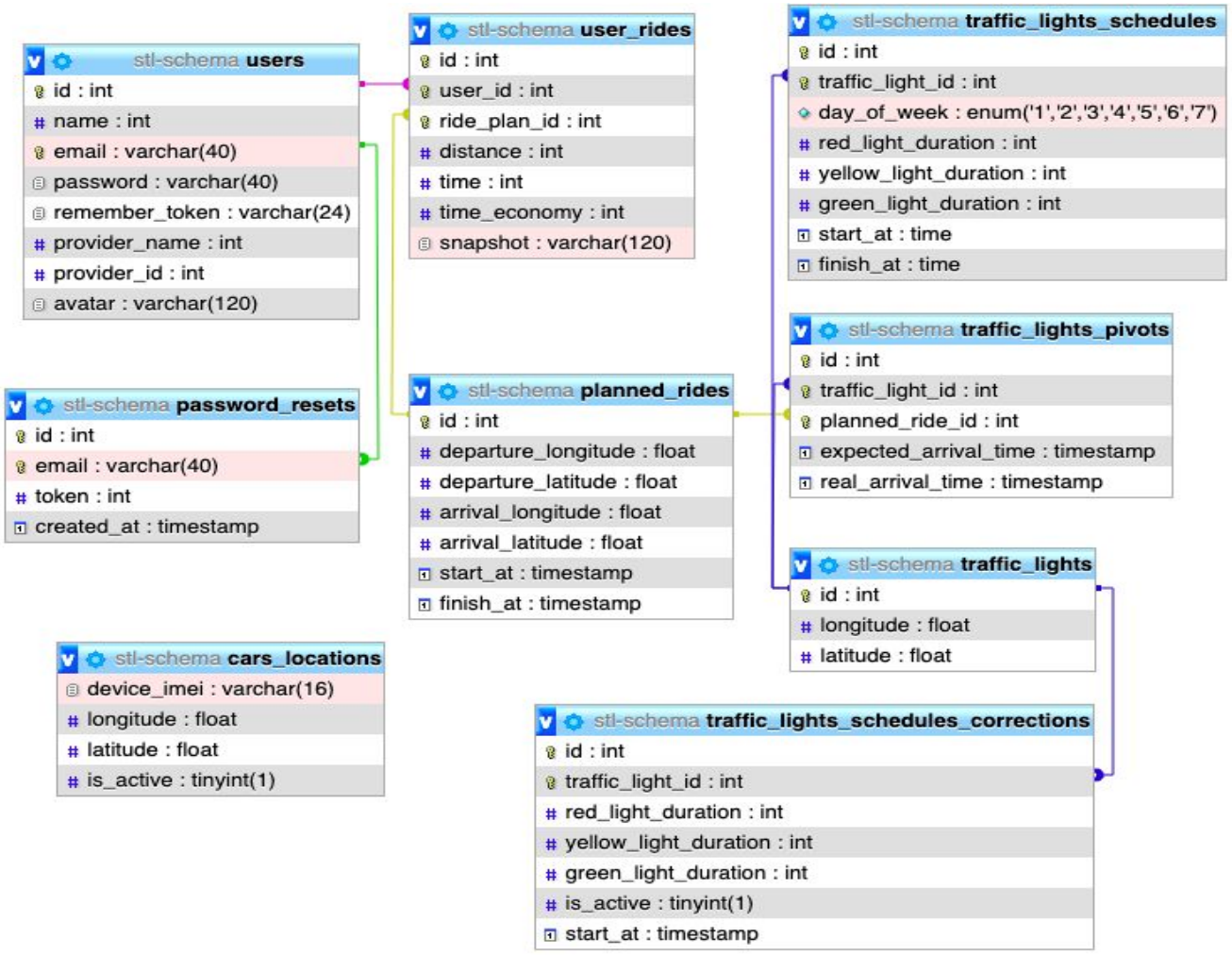

Рисунок 4.3 — Схема бази даних системи управління транспортними потоками

База даних для відображення фізичної моделі містить наступні таблиці: “cars_locations”, “traffic_lights”, “traffic_lights_pivots”, “traffic_lights_schedules”, “traffic_lights_schedules_corrections”, “planned_rides” та “user_rides”.

Завдяки такій особливості проектування бази даних стало можливим також і отримання цієї інформації зі сторонніх ресурсів, тим самим збільшуючи вибірку, на якій грунтується розрахунок трафіку і зменшуючи похибку таких розрахунків.

В таблиці “cars_locations” зберігається інформація, що складає основу для розрахунків моделі транспортних заторів — кількість, геолокація та швидкість пересування автомобільних транспортних засобів (таблиця 4.1).

Таблиця 4.1 — Структура таблиці БД “cars_locations”

Поле	**Тип даних**	**Функціональне призначення**
device_imei	string (16)	Унікальний серійний ідентифікаційний номер персонального кишенькового пристрою
longitude	float	Довгота місця розташування пристрою
latitude	float	Широта місця розташування пристрою
is_active	bool	Поле-перемикач, яке вказує, чи ділиться власник пристрою своєю геолокацією в даний момент. Якщо значення false, система зберігає останнє відоме місце розташування.

Не важко помітити, що це єдина таблиця, котра не має зв’язків з іншими таблицями бази даних, і це не випадковість. Така особливість пов’язана з тим, що геолокація є персональною інформацією і піддається закону про охорону конфіденційності персональних даних. Виходячи з цих міркувань, інформація не прив’язується до акаунтів користувачів web-системи, забезпечуючи анонімність, а в якості унікального ключа використовується IMEI.

В якості альтернативного джерела інформації про завантаженість автомобільних доріг використовується також набір RESTful API сервісів (TomTom Traffic Stats, Google Maps Javascript API, Google Geocoding API, Here Technologies API), який аналізує та віддає історичну інформацію про автомобільний трафік, володіючи доступом до мільйонів навігаційних пристроїв та вбудованих систем в режимі реального часу.

Таблиця під назвою “traffic_lights” зберігає інформацію про інший, не менш важливий інформаційний об’єкт — розумний світлофор, який оснащено інтернет-модулем та наділено можливістю приймати через мережевий канал сигнали на зміну тривалості циклів переключення кольорів (таблиця 4.2).

Таблиця 4.2 — Структура таблиці БД “traffic_lights”

Поле	**Тип даних**	**Функціональне призначення**
id	int	Автоінкремент з унікальним для таблиці значенням
longitude	float	Довгота геолокації світлофора
latitude	float	Широта геолокації світлофора

Крім місця розташування світлофорів зберігаються також їх звичайні графіки циклів (таблиця 4.3).

Таблиця 4.3 — Структура таблиці БД “traffic_light_shedules”

Поле	**Тип даних**	**Функціональне призначення**
id	int	Автоінкрементний ідентифікатор
traffic_light_id	int	Ідентифікатор світлофора
day_of_week	enum	Числові представлення днів тижня (від 1 до 7)
red_light_duration	int	Тривалість червоного світла, сек

yellow_light_duration	int	Тривалість жовтого світла, сек
green_light_duration	int	Тривалість зеленого світла, сек
start_at	timestamp	Початок роботи світлофора за таким графіком
finish_at	timestamp	Час завершення роботи світлофора за поточним графіком

База даних також містить таблицю “traffic_light_shedules_corrections”, яка активно оновлюється актуальними циклами роботи через заздалегідь визначені проміжки часу (таблиця 4.4). Така можливість програмування відкладених подій реалізована через механізм Laravel під назвою черги.

Таблиця 4.4 — Структура таблиці БД “traffic_light_shedules_corrections”

Поле	**Тип даних**	**Функціональне призначення**
id	int	Автоінкрементний ідентифікатор
traffic_light_id	int	Ідентифікатор світлофора
red_light_duration	int	Тривалість червоного світла, сек
yellow_light_duration	int	Тривалість жовтого світла, сек
green_light_duration	int	Тривалість зеленого світла, сек
start_at	timestamp	Початок роботи світлофора за таким графіком
is_active	bool	Показник активності внесених до графіку змін

Якщо користувач авторизувався, проклав шлях з пункту А в пункт Б і підтвердив здійснення поїздки запропонованим маршрутом, вона збережеться у системі і в подальшому буде використана для розрахунку оцінки завантаженості світлофорів (таблиця 4.5).

Таблиця 4.5 — Структура таблиці БД “planned_rides”

Поле	Тип даних	Функціональне призначення
id	int	Автоінкрементний ідентифікатор
departure_longitude	float	Довгота пункту відправки
departure_latitude	float	Широта пункту відправки
arrival_longitude	float	Довгота пункту призначення
arrival_latitude	float	Широта пункту призначення
start_at	timestamp	Час початку поїздки
finish_at	timestamp	Час завершення

Оскільки для розрахунку ефективності алгоритму керування транспортними потоками маршрут розбивається світлофорами на сегменти, база даних містить уточнюючу таблицю для збереження проміжних показників (таблиця 4.6).

Таблиця 4.6 — Структура таблиці БД “traffic_light_pivots”

Поле	Тип даних	Функціональне призначення
id	int	Автоінкрементний ідентифікатор
traffic_light_id	int	Ідентифікатор світлофора
planned_ride_id	enum	Ідентифікатор плану поїздки
expected_arrival_time	int	Очікувана тривалість проходження маршруту без алгоритму, сек
real_arrival_time	int	Фактична тривалість проходження маршруту із алгоритмом, сек

Відкриваючи історію поїздок можна буде пригадати пункти відправки та призначення, довжину маршруту та час, який було зекономлено завдяки сервісу розумного керування транспортними потоками, а збережений скріншот нагадає, якими автомобільними дорогами пролягав шлях (таблиця 4.7).

Таблиця 4.7 — Структура таблиці БД “user_rides”

Поле	**Тип даних**	**Функціональне призначення**
id	int	Автоінкрементний ідентифікатор
user_id	int	Ідентифікатор користувача
ride_plan_id	int	Ідентифікатор плану поїздки
distance	int	Довжина маршруту, м
time	int	Час, проведений в дорозі, сек
time_economy	int	Час, зекономлений у поїздці завдяки алгоритму управління транспортними потоками
snapshot	string (160)	Посилання на знімок карти із прокладеним на ній маршрутом

Зібрані дані дають системі можливість розрахувати, в який момент часу на кожному з керованих розумних світлофорів утвориться пікове навантаження і згладити цю криву, розіславши світлофорам сигнали на зміну циклу поведінки.

4.4. Діаграма прецедентів

Діаграма прецедентів UML — це діаграма, що відображає відносини між акторами та прецедентами і є складовою частиною моделі прецедентів, що дозволяє

на концептуальному рівні здійснити детальний опис системи (рисунок 4.4).

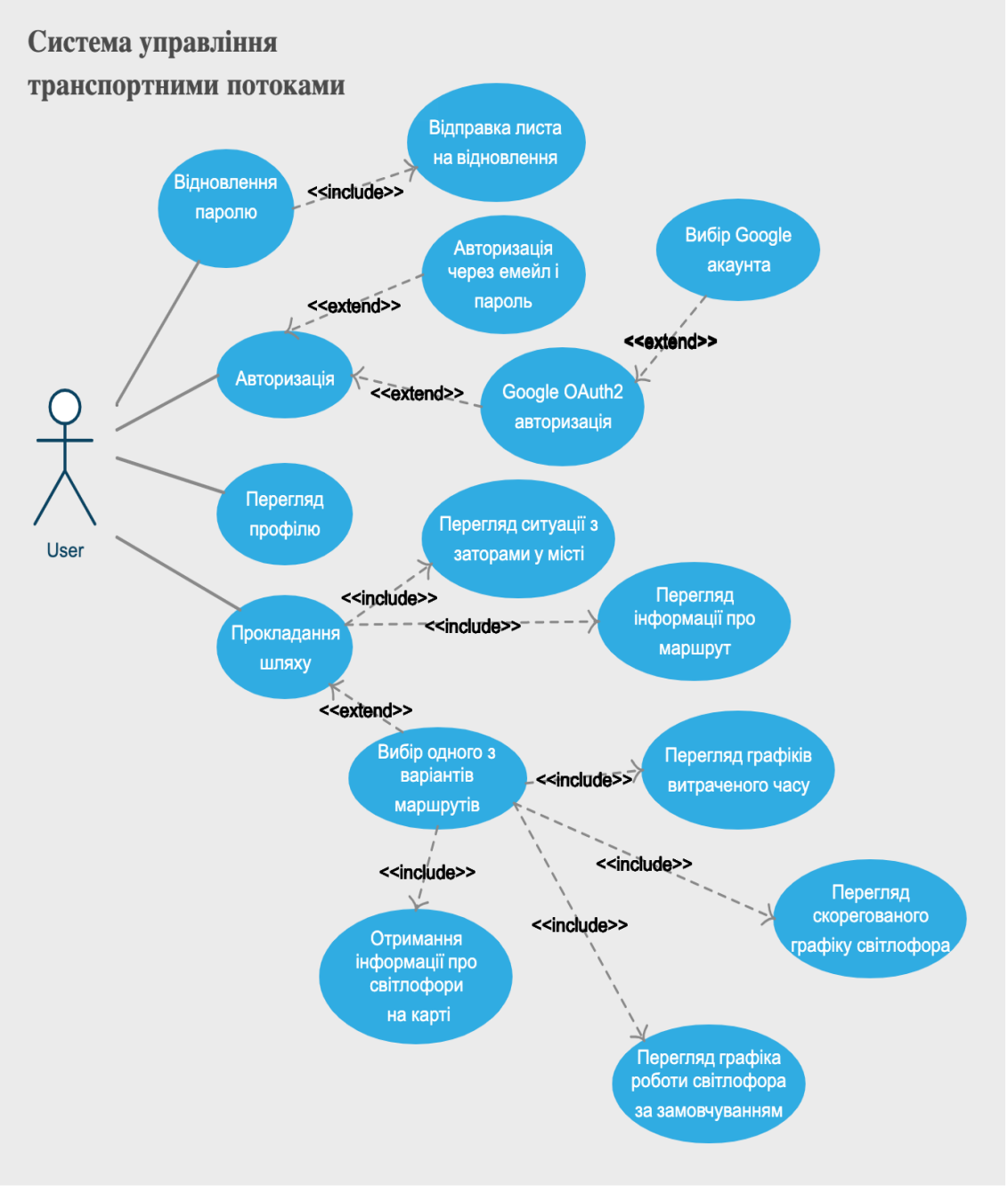

Рисунок 4.4 —Діаграма прецедентів

Прецедентом є функція-частина змодельованої системи, використовуючи яку актор на виході отримує конкретний результат, котрий задовольняє поставленим вимогам. В якості прецеденту, як правило, обирають відокремлений сервіс, блок функціоналу який є одним з типових варіантів використання системи користувача,

прикладною точкою їхньої взаємодії. Прецеденти можуть слугувати допоміжними елементами при формування специфікації зовнішніх вимог.

Для відображення моделі прецедентів на діаграмі використовуються такі символи:

— рамки системи — прямокутник з назвою у верхній частині і еліпсами (прецедентами) всередині;

— актор — символ людини, що позначає набір ролей користувача (розуміється в широкому сенсі: людина, зовнішня сутність, клас, інша система);

— прецедент — еліпс з написом, що позначає виконувані системою дії, що призводять до спостережуваних акторами результатів. Напис може бути ім'ям або описом (з точки зору акторів) того, що робить система. Ім'я прецеденту пов'язано з неперервним сценарієм – конкретної послідовністю дій, що ілюструє поведінку. В ході сценарію актори обмінюються з системою повідомленнями. Сценарій може бути приведений на діаграмі прецедентів у вигляді UML-коментаря. З одним прецедентом може бути пов'язано кілька різних сценаріїв.

4.5. Висновки до розділу

Розділ проектування web-сервісу містить схеми, діаграми та графіки, які репрезентують технічне завдання та слугують прототипами (лекалами), за якими на наступному етапі велося безпосереднє написання коду на мовах програмування PHP та JavaScript.

Створена база даних описує транспортну екосистему, тобто фізичну модель дипломної роботи. БД зберігає таблиці з користувацькою інформацією, геолокацією

автомобілів, місцем розташування та графіками роботи доступних для перепрограмування розумних світлофорів та виправлення до цих графіків, які слугують вхідними даними для алгоритму виявлення світлофорів з максимальним навантаженням та побудови для них нового розкладу на піковому проміжку часу.

Архітектура системи, детально проілюстрована візуально, описує вхідні потоки даних (метрики з GPS-навігаторів), які обробляються і перетворюються в моделі трафіку та базу знань про активні маршрути, в свою чергу вони після пошуку розумних світлофорів на шляху та отримання даних про їх завантаженість в момент часу x беруть участь у алгоритмах виявлення точок з навантаженням та побудовою оновлених графіків роботи світлофорів в години пік. Трансформовані графіки виводяться користувачеві через браузер на екран і можуть бути передані як JSON-файл на регульований світлофор. За схемою, яку він містить цей світлофор може бути перепрограмовано на апаратному рівні.

Розроблену систему було описано за допомогою UML діаграми варіантів використання, на якій відображено можливі шляхи застосування користувачем змодельованого додатку. Прецедентами виступають такі функції програми, як авторизація через логін та пароль або OAuth2 протокол та персональний Google аккаунт, відновлення доступу до системи через скидання паролю, перегляд власного профілю та стандартних графіків роботи світлофорів, передача геолокації, прокладання маршруту в пункт призначення з можливістю переглянути ситуацію по заторам у місті, деталізацію обраного маршруту, перегляд скоригованого графіку світлофора, якщо такий трапився на вибраному маршруті ти підсумкові графіки, які наочно демонструють вигоду від використання web-додатку.

5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Створення web-сервісу відбувається з метою реалізувати GLOSA методологію та впровадити її широке застосування на мобільних пристроях, що перебувають на борту автомобільних транспортних засобів, тим самим втілюючи в життя переваги такого підходу до вирішення проблем із заторами на дорогах, включаючи небезпечне для навколишнього середовища та економічно не вигідне надмірне спалювання топлива, збільшення тривалості поїздок через необхідність чекати зелене світло світлофора, підвищення небезпеки автомобільних катастроф тощо.

Продукт повинен мати виділений сервер і не використовувати ресурси кінцевого користувача для важких розрахунків та алгоритмів.

Оскільки продукт є web-сервісом, мінімальна конфігурація комп’ютера відповідає вимогам до браузера, через який користувач взаємодіє з програмним забезпеченням. Вимоги до програмної сумісності відсутні з аналогічних причин.

5.1. Опис функцій та вигляду розробленої системи

Взаємодія кінцевого користувача з web-сервісом розпочинається з авторизації. Система надає можливість обирати спосіб входу в систему (рисунок 5.1).

Перша опція — вхід через Google account. Google Apps використовує найновіший авторизаційний стандарт OAuth2 і двофакторну авторизацію, якщо така увімкнена в обліковому записі G Suite. Альтернативний спосіб входу більш консервативний, через електронну поштову скриньку та пароля. Варто зауважити,

що форма реєстрації в систему відсутня, оскільки цю функцію бере на себе форма логіну, якщо вказана пошта використовується для входу у систему вперше.

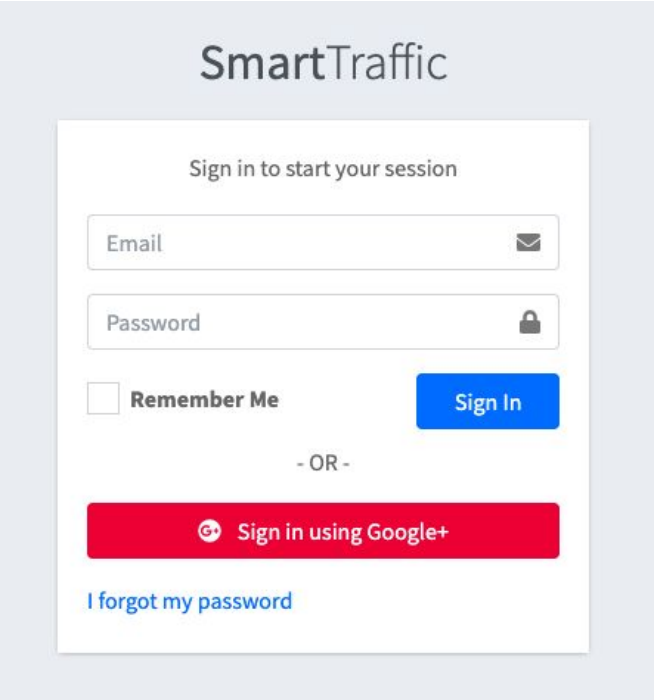

Рисунок 5.1 — Вікно авторизації в систему

Натомість, у випадку, якщо користувач забуде свій пароль, йому доступна можливість відновити його, перейшовши за посиланням “Forgot Your Password?”, розташованим внизу форми. Натискання на цю кнопку переспрямує користувача на форму для відновлення паролю (рисунок 5.2).

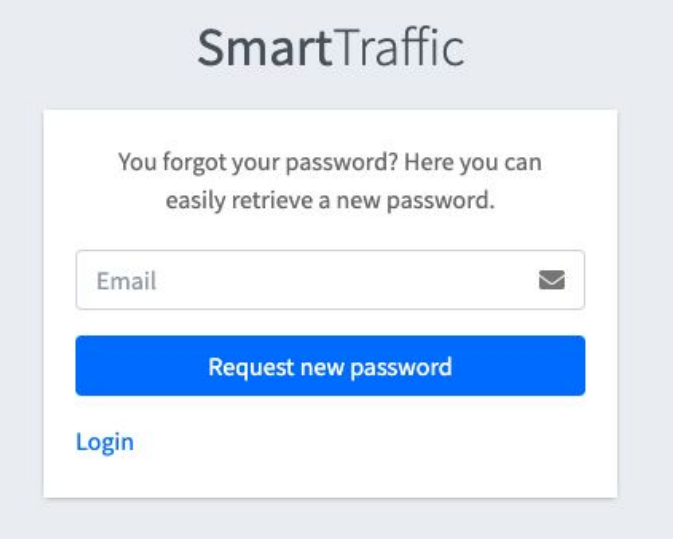

Рисунок 5.2 — Форма відновлення паролю

У випадку, якщо перехід було здійснено випадково, або користувач пригадав, що вже зареєстрований у системі, завжди можна повернутись у попереднє авторизаційне вікно, скориставшись посиланням “Login” внизу форми.

Інтерфейс системи для керування транспортними потоками складається з бічного меню, верхнього горизонтального меню та основного вікна програми.

З горизонтального меню доступна можливість вийти із системи, а при натисканні на піктограму, розташовану в лівому кутку горизонтального меню, ширина бічної панелі зменшиться, підписи пунктів меню зникнуть і панель перетвориться на лаконічну вузьку полоску з піктограмами, при наведенні вказівником на які випливуть підказки.

При натисканні на логотип web-сервісу користувач перейде на домашню сторінку, контент якої буде описано далі. Для перегляду профіля, стандартних або відкоригованих графіків циклів переключень світлофорів, переходу до функціоналу управління транспортними потоками можна скористатися відповідними пунктами меню на бічній панелі.

В ролі початкової сторінки web-сервісу виступає Dashboard (рисунок 5.3). Плашки на верхній частині основного вікна системи відображають коротку статистику про:

— Лічильник маршрутів, які користувач будував через систему;

— Кількість користувачів (мобільних пристроїв, що мають вбудований GPS навігатор), які діляться інформацією про своє місце розташування, тим самим виступаючи у ролі метрики для прогнозування заторів на дорогах;

— Число розумних світлофорів у системі, цикли перемикання яких можна регулювати, надсилаючи відповідні сигнали;

— Інформація про поточну ситуацію з автомобільними заторами (оцінка від 1 до 10) для місцевості, що відповідає геолокації користувача.

Остання плашка може змінювати свій колір по шкалі від зеленого до червоного, грунтуючись на оцінці по шкалі заторів.

Градація кольорів показана під розділом статистики.

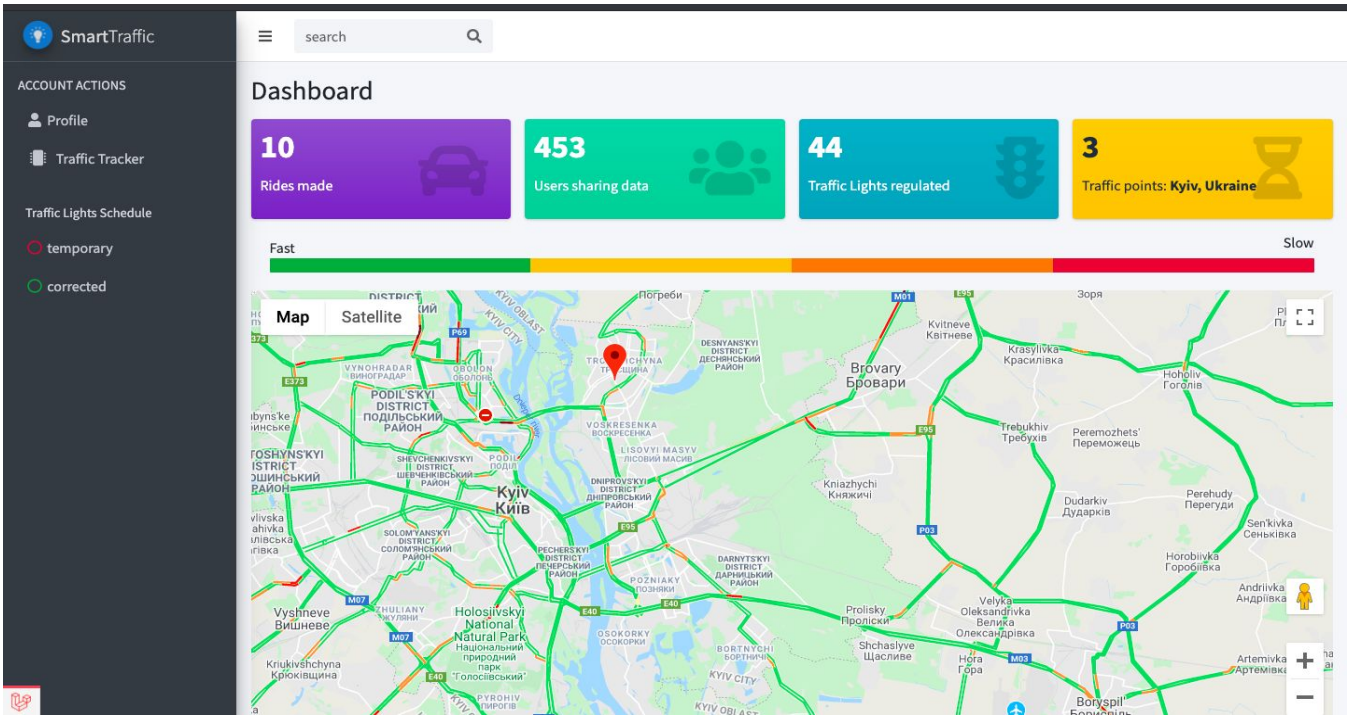

Рисунок 5.3 — Початкова сторінка додатку

Останнім елементом на цій сторінці є мапа. Користувачу доступна можливість змінювати її масштаб і розгортати на всю ширину вікна браузера. При бажанні користувач може змінити тип відображення мапи на супутникові знімки і перейти у режим перегляду вулиць. Якщо система втратила сесію або користувач користується нею вперше, він отримає випливаюче віконце з пропозицією поділитися своєю локацією (рисунок 5.4). На основі цієї інформації на карті нижче проставиться мітка, яка стане її центром, тим самим відображаючи карту околиць.

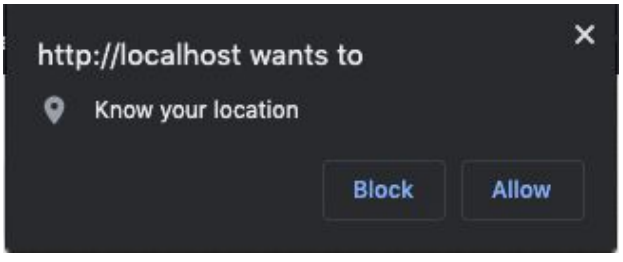

Рисунок 5.4 — Вікно запиту дозволу на отримання місця розташування

Сторінка Traffic tracker має інтуїтивно зрозумілий інтерфейс (рисунок 5.5). Для запуску алгоритмів трекера достатньо просто ввести пункт відправки в дорогу

та пункт кінцевого призначення. Зверху над цими полями розташована підказка, яка розгортається при наведенні і дає користувачу інформацію про те, як працює система і яку користь вона приносить. Після натискання на кнопку “GO” система почне шукати варіанти маршрутів, їх протяжність та середній час їзди із врахуванням проблем на дорогах.

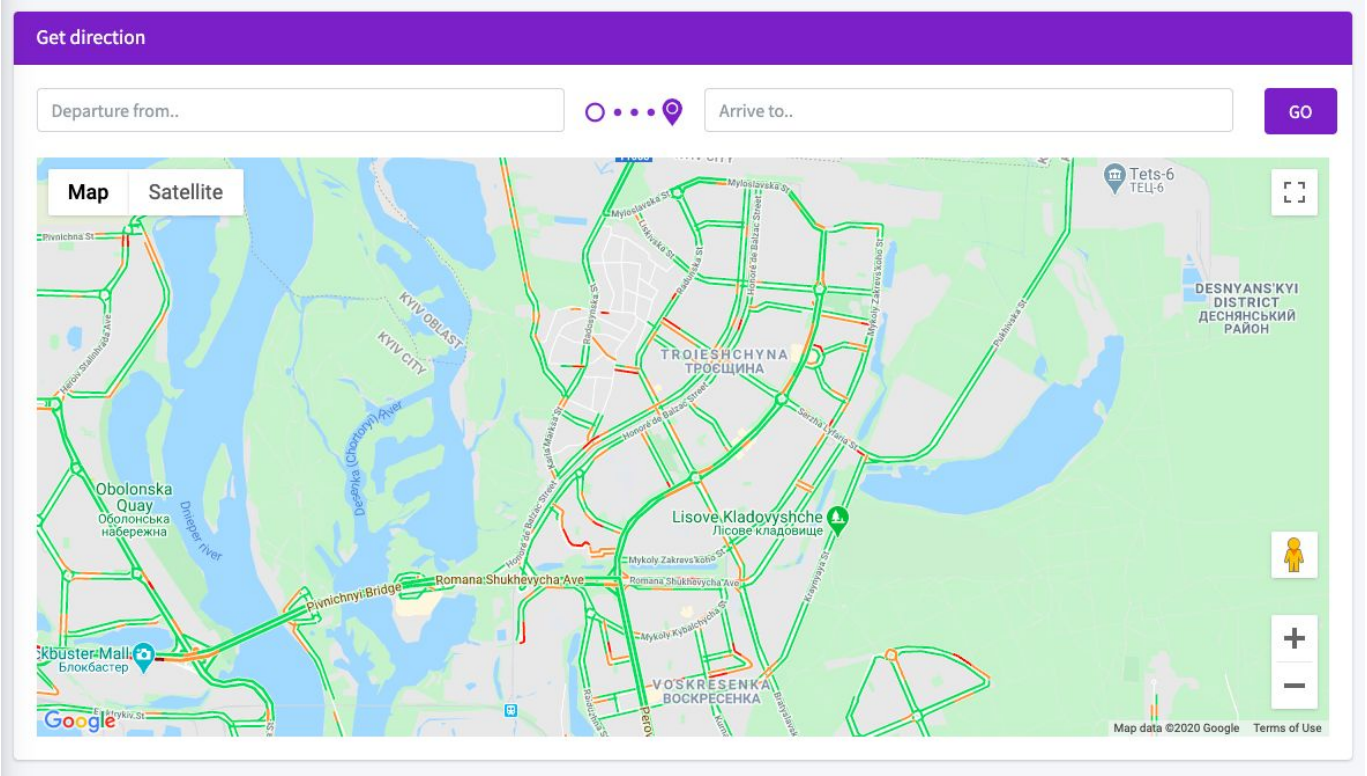

Рисунок 5.5 — Початкова сторінка Traffic Tracker

Користувачу також буде запропоновано обрати один з маршрутів. Після цього система почне розрахунки, результати яких буде відображено в окремій вкладці браузера (рисунок 5.6).

Форма, візуально розділена навпіл, на першій частині відмалює мапу, де будуть показані пункти відправки та прибуття і маршрут проїзду між ними. Якщо на цьому шляху зустрічаються світлофори, доступом до регулювання яких володіє система, їх також буде позначено мітками на карті.

Друга частина форми доносить користувачу інформацію з деталями обраного шляху поїздки (протяжність маршруту та очікуваний час в дорозі).

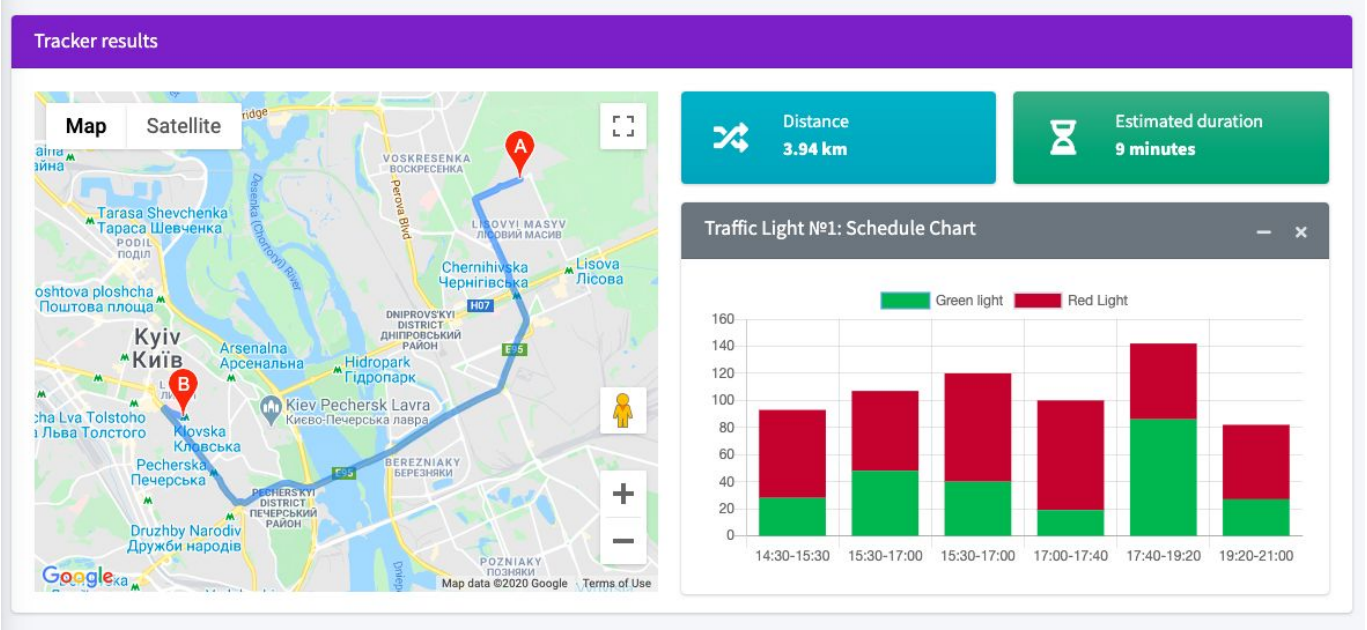

Рисунок 5.6 — Результуюча сторінка Traffic Tracker

Також сервіс видасть декілька результуючих таблиць і графіків. Для світлофорів, які лежать на обраному маршруті створиться графік, на якому буде показано послідовність переключень його циклів. Якщо система на основі алгоритму зареєструє піковість трафіку на регульованому світлофорі, його розклад буде скориговано, і це відобразиться на відповідному графіку. У вигляді графіку користувачу також нададуть порівняння швидкостей прибуття в місце призначення з та без корекції змін в графіки світлофорів.

5.2. Висновки до розділу

В останньому розділі звіту було викладено короткий підсумок створеної системи та вичерпний опис її інтерфейсу з послідовністю дій, яких потрібно дотримуватися для проходження по кожній функції сервісу.

ВИСНОВКИ

В рамках виконання завдання на дипломну роботу, крім визначення та ознайомлення на практиці із засобами та інструментами, що утворять технічну базу для дипломної роботи, дослідження літератури по темі розумного керування транспортними потоками, налагодження інфраструктури проекта, проектування бази даних та UML діаграми варіантів використання було вирішено наступні задачі:

1. Отримано геолокацію користувача системи, з її використанням намальовано мітку на карті, накладено шар транспортної ситуації та розраховано оцінку заторів в місті перебування користувача за 10-бальною шкалою;
2. Розроблено навігаційну форму Traffic Tracker для прокладання маршруту з пункту відправки в пункт призначення;
3. Зібрано в ручну дані про місце розташування та цикли розумних світлофорів, внесено їх в БД;
4. Зібрано та проаналізовано трафік на дорогах міста;
5. Розроблено алгоритми для виявлення регульованих світлофорів на шляху та розрахунку його завантаженості;
6. Відфільтровано світлофори з піковим навантаженням та побудовано нові розклади для них в години-пік;
7. Реалізовано побудову маршруту, відображення відміток регульованих світлофорів на ньому, відформатовано та подано інформацію про відстань та середній час подолання шляху, відображено старі та оптимізовані графіки світлофора, якщо вони потребували зміни;
8. Побудовано математичні графіки для візуального представлення ефективності роботи алгоритму по розумному керуванню транспортними потоками;
9. Зібрано додаткові метрики для формування дашборда.

Перед початком безпосереднього процесу розробки було створено технічне завдання, спроектовано базу даних і описано варіанти використання програмного продукту через UML діаграму, підготовано інфраструктуру на базі MySQL, phpMyAdmin, nginx та ubuntu контейнерів Docker. Проведено відбір технологій та середовища розробки програмного забезпечення, з використанням яких створено проектну частину дипломної роботи. При розробці було успішно застосовано архітектурний патерн MVC. В процесі було закріплено отримані під час лекційних та практичних занять з web-програмування.

Підсумовуючи результати проходження переддипломної практики та беручи до уваги теоретичні знання та практичні навички, набуті в процесі можна сказати, що поставлені задачі було успішно виконано.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 2019 URBAN MOBILITY REPORT [Електронний ресурс] // The Texas A&M Transportation Institute with cooperation from INRIX. – 50 с. – Режим доступу до ресурсу: https://static.tti.tamu.edu/tti.tamu.edu/documents/mobility-report-2019.pdf
2. 1966 Traffic Signals,. Webster, F.V., Cobbe, B.M. // London:Road Research Technical Paper N56, HMSQ. – 111 с.
3. 2011 SignalGuru: Leveraging Mobile Phones for Collaborative Traffic Signal Schedule Advisory, E. Koukoumidis, L. Peh, M. Martonosi. [Електронний ресурс] – 14 с. – Режим доступу до ресурсу: https://mrmgroup.cs.princeton.edu/papers/Koukoumidis_SignalGuru_MobiSys_2011.pdf
4. 2017 Procedure for Calculating On-Time Duration of the Main Cycle of a Set of Coordinated Traffic Lights, V. Gorodokin, Z. Almetova, V. Shepelev [Електронний ресурс] // Transportation research Procedia. – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S2352146517300601#bbib0060
5. 2015 Расчет промежуточного такта цикла работы светофорного объекта, В. Городкин, З. Альметова, О. Леонова. [Електронний ресурс] // Вестник СибАДИ. – Режим доступу до ресурсу: https://cyberleninka.ru/article/n/raschet-promezhutochnogo-takta-tsikla-raboty-svetofornogo-obekta/viewer
6. 2019 Laravel: Up & Running: A Framework for Building Modern PHP Apps, M. Stauffer // O'Reilly Media, 2 edition. – 554 с.
7. 2017 Building a responsive web application with the MVC PHP framework, J. Anttonen // Lahti University of Applied Sciences. – 71 с.
8. 2016 Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5, 4-е издание, Р. Никсон . – 768 с.

9. 2017 Mastering Docker, R. McKendrick, S. Gallagher // Packt Publishing, 2 ed. – 392 с.
10. 2009 An Introduction to Relational Database Theory, H. Darwen // bookboon.com. – 235 с.
11. 2015 SQL. Полное руководство, Д. Р. Грофф, П.Н. Вайнберг, Э. Дж. Оппель // Вильямс . – 960 с.
12. https://www.uml.org/what-is-uml.htm

ДОДАТОК А

Інструментальні засоби управління транспортними потоками

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ6168_20Б

Аркушів 2

Київ — 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ6168_20Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ6168_20Б 12-1	geo.blade.php	Компонент для отримання геолокації, формування мапи, накладання шару транспортних заторів
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ6168_20Б 12-2	SmartTrafficLightsOptimisationService.php	Компонент для формування оптимізованого графіку переключення світлофора

ДОДАТОК Б

Інструментальні засоби управління транспортними потоками

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ6168_20Б 12-1

Аркушів 11

Київ — 2020

geo.blade.php

```
@section('js')
<script src="https://maps.googleapis.com/maps/api/js?key={{ $apiKey }}"></script>

<script>
  const initMap = ({latitude, longitude}) => {
    const currentGeolocation = {
      lat: latitude,
      lng: longitude,
    };
    const parameters = {
      zoom: 13,
      center: currentGeolocation,
    };
    const map = new google.maps.Map(document.getElementById('map'), parameters);
    const trafficLayer = new google.maps.TrafficLayer();
    trafficLayer.setMap(map);
    const marker = new google.maps.Marker({position: currentGeolocation, map: map});
  };
  const getLocation = () => {
    if (navigator.geolocation) {
      navigator.geolocation
        .getCurrentPosition(({coords}) => {
          const {latitude, longitude} = coords;
          initMap({latitude, longitude})
        });
    }
  }
  getLocation();

  const calcCityData = ({latitude, longitude}) => {
    latlng = new google.maps.LatLng(latitude, longitude);
```

```
new google.maps.Geocoder().geocode({'latLng' : latlng}, function(results, status) {
   if (status == google.maps.GeocoderStatus.OK) {
      if (results[1]) {
         var country = null, countryCode = null, city = null, cityAlt = null;
         var c, lc, component;
         for (var r = 0, rl = results.length; r < rl; r += 1) {
            var result = results[r];

            if (!city && result.types[0] === 'locality') {
               for (c = 0, lc = result.address_components.length; c < lc; c += 1) {
                  component = result.address_components[c];

                  if (component.types[0] === 'locality') {
                     city = component.long_name;
                     break;
                  }
               }
            }
            else if (!city && !cityAlt && result.types[0] === 'administrative_area_level_1') {
               for (c = 0, lc = result.address_components.length; c < lc; c += 1) {
                  component = result.address_components[c];

                  if (component.types[0] === 'administrative_area_level_1') {
                     cityAlt = component.long_name;
                     break;
                  }
               }
            } else if (!country && result.types[0] === 'country') {
               country = result.address_components[0].long_name;
               countryCode = result.address_components[0].short_name;
            }
```

```
                if (city && country) {
                    break;
                }}

            document.getElementById('city').innerHTML = city + ', ' + country;
        }}});
    }

    if (navigator.geolocation) {
        navigator.geolocation
            .getCurrentPosition(({coords}) => {
                const {latitude, longitude} = coords;
                calcCityData({latitude, longitude})
            });
    }
</script>

<script>
    function initMap() {
        var directionsService = new google.maps.DirectionsService();
        var directionsRenderer = new google.maps.DirectionsRenderer();
        var map = new google.maps.Map(document.getElementById('map'), {
            zoom: 7,
            center: {lat: document.getElementById('fglat').value, lng:
document.getElementById('fglon').value}
        });

        directionsRenderer.setMap(map);
        calculateAndDisplayRoute(directionsService, directionsRenderer);
    }

    initMap();
```

```
  function calculateAndDisplayRoute(directionsService, directionsRenderer) {
    directionsService.route(
      {
        origin: {query: document.getElementById('from').value},
        destination: {query: document.getElementById('to').value},
        travelMode: 'DRIVING'
      },
      function (response, status) {
        if (status === 'OK') {
          directionsRenderer.setDirections(response);
        } else {
          window.alert('Directions request failed due to ' + status);
        }});
 }
</script>
```

SmartTrafficLightsOptimisationService.php

```
<?php

use App\GeolocationPoint;
use App\Services\TomTom\TrafficFlowApi\TrafficFlowApi as TomTomTrafficService;
use App\Services\Google\TrafficFlowApi\TrafficFlowApiService as GoogleTrafficService;
use App\TrafficLight;
use App\TrafficLightScheduleCorrection;
use App\Repositories\TrafficLightScheduleCorrectionRepository;
use Carbon\Carbon;
use Illuminate\Support\Collection;

class SmartTrafficLightsOptimisationService
{
  /** @var TomTomTrafficService */
  private $tomTomTrafficService;

  /** @var GoogleTrafficService */
  private $googleTrafficService;

  /** @var TrafficLightCycleFormatter */
  private $cycleFormatter;

  /** @var TrafficLight */
  private $trafficLight;

  /** @var TrafficLightScheduleCorrectionRepository */
  private $trafficLightScheduleCorrectionRepository;

  /** @var AlgoythmDurationCalculator */
  private $algorythmDurationCalculator;

  /** @var int */
```

```
private $linesCount;

/** @var float */
private $lineWidth;

/** @var string */
private $waitingTransportGeneralizedSchema;

/** SmartTrafficLightsOptimisationService constructor.
 *
 * @param TomTomTrafficService $tomTomTrafficService
 * @param GoogleTrafficService $googleTrafficService
 * @param TrafficLightCycleFormatter $trafficLightCycleFormatter
 * @param TrafficLightScheduleCorrectionRepository $trafficLightScheduleCorrectionRepository
 * @param AlgoythmDurationCalculator $algorythmDurationCalculator
 * @param string $waitingTransportGeneralizedSchema
 * @param int $trafficLightId
 * @param int $linesCount
 * @param null|float $lineWidth
 */
public function __construct(
    TomTomTrafficService $tomTomTrafficService,
    GoogleTrafficService $googleTrafficService,
    TrafficLightCycleFormatter $trafficLightCycleFormatter,
    TrafficLightScheduleCorrectionRepository $trafficLightScheduleCorrectionRepository,
    AlgoythmDurationCalculator $algorythmDurationCalculator,
    string $waitingTransportGeneralizedSchema,
    int $trafficLightId,
    int $linesCount,
    ?float $lineWidth
) {
    $this->tomTomTrafficService = $tomTomTrafficService;
    $this->googleTrafficService = $googleTrafficService;
```

```
        $this->cycleFormatter = $trafficLightCycleFormatter;
        $this->trafficLightScheduleCorrectionRepository = $trafficLightScheduleCorrectionRepository;
        $this->algorythmDurationCalculator = $algorythmDurationCalculator;

        $this->waitingTransportGeneralizedSchema = json_decode($waitingTransportGeneralizedSchema,
true);

        $trafficLightsCollection = TrafficLight::all();
        $this->trafficLight = $trafficLightsCollection->where('id', $trafficLightId);

        $this->linesCount = $linesCount;
        $this->lineWidth = $lineWidth ?? 3;
    }

    /**
     * @param float|null $speed
     * @param float|null $decelation
     *
     * @return array
     */
    public function optimize(?float $speed, ?float $decelation): array
    {
        $transportCycleDuration = $this->transportCycleDuration($speed, $decelation);
        $pedestrianCycleDuration = $this->pedestriansCycleDuration();

        $duration = $this->algorythmDurationCalculator->calcDuration();

        $result = $this->trafficLightScheduleCorrectionRepository->updateTable(
            $transportCycleDuration,
            $pedestrianCycleDuration,
            $duration
        );
```

```
    return $this->cycleFormatter->format($result);
}

private function transportCycleDuration(
    float $speed = 11.11,
    float $decelation = 4.6,
    float $reactionTime = 0.6,
    float $engineDelay = 0.1,
    float $slowdown = 0.35
): int
{
    $singeCycleTransportFlowVolume = $this->calculateSingeCycleTransportFlowVolume();

    $enterDistance = $this->calculateEnterDistance($this->waitingTransportGeneralizedSchema);

    $midt = $reactionTime + $engineDelay + 0.5 * $slowdown;
    $distanceToWait = $midt * ($speed / 3.6) + pow(2, $speed) / (26 * $decelation);

    $delayToStart = collect($this->waitingTransportGeneralizedSchema)->count() * 1.5;

    return sqrt((2 * ($enterDistance - $distanceToWait)) / 1.5) + $delayToStart;
}

/**
 * @return float
 */
private function calculateSingeCycleTransportFlowVolume(): float
{
    /** @var GeolocationPoint $trafficLightGeolocation */
    $trafficLightGeolocation = $this->trafficLight->getGeolocation();

    $currentTimestamp = (int)Carbon::now()->toDateTimeString();
    $hourAgoTimestamp = (int)Carbon::now()->subHour()->toDateTimeString();
```

```
    $currentDay = Carbon::now()->dayOfWeek;

    $googleHourActivity = $this->googleTrafficService->getActivity(
        $hourAgoTimestamp,
        $currentTimestamp,
        $trafficLightGeolocation);

    $tomTomHourActivity = $this->tomTomTrafficService->getHistoricalActivity(
        $currentDay,
        $hourAgoTimestamp,
        $currentTimestamp,
        $trafficLightGeolocation);

    $previousHourActivity = $googleHourActivity * 0.7 + $tomTomHourActivity * 0.3;

    return ($previousHourActivity * 120) / 3600;
}

/**
 * @param array $waitingTransportSchema
 * @param float $avgDistance
 * @param float $avgTransportLength
 *
 * @return int
 */
private function calculateEnterDistance(
    array $waitingTransportSchema,
    float $avgDistance = 1,
    float $avgTransportLength = 4.5
): int
{
    $waitingTransportSchema = new Collection($waitingTransportSchema);
```

```
    $count = $waitingTransportSchema->count();

    return ($count > 10)
        ? $this->realEnterDistance($waitingTransportSchema, $count)
        : $count * ($avgDistance + $avgTransportLength);
}

/**
 * @param Collection $waitingTransportSchema
 * @param int $count
 *
 * @return float
 */
private function realEnterDistance(Collection $waitingTransportSchema, int $count): float
{
    $distance = $count - 1;

    $bikesCount = $waitingTransportSchema->where('type', 'bike')->count();
    $motorcyclesCount = $waitingTransportSchema->where('type', 'motorcycle')->count();
    $minibusCount = $waitingTransportSchema->where('type', 'minibus')->count();
    $carCount = $waitingTransportSchema->where('type', 'car')->count();
    $busCount = $waitingTransportSchema->where('type', 'bus')->count();
    $trolleyBusCount = $waitingTransportSchema->where('type', 'trolleybus')->count();

    return (
        $bikesCount * TransportLengthInterface::BIKE +
        $motorcyclesCount * TransportLengthInterface::MOTO +
        $minibusCount * TransportLengthInterface::MINIBUS +
        $carCount * TransportLengthInterface::CAR +
        $busCount * TransportLengthInterface::BUS +
        $trolleyBusCount * TransportLengthInterface::TROLLEYBUS +
        $distance
    );
```

```
    }

    /**
     * @param float $pedestrianSpeed
     *
     * @return int
     */
    private function pedestriansCycleDuration($pedestrianSpeed = 1.3): int
    {
        $pedestrianDistance = $this->linesCount * $this->lineWidth;

        return 5 + ($pedestrianDistance / $pedestrianSpeed);
    }
}
```

ДОДАТОК В

Інструментальні засоби управління транспортними потоками

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ6168_20Б 13-1

Аркушів 9

Київ — 2020

АНОТАЦІЯ

Додаток містить опис web-додатку для керування транспортними потоками з метою оптимізації ситуації з заторами на дорогах і виконує завдання, зазначені в розділі 1, а саме:

— Отримання геолокації користувача, її мітка та затори в в околицях на мапі;
— Розробка форми для прокладання користувачем маршруту;
— Збір даних про місце розташування і графік переключень розумних світлофорів та внесення їх в базу даних системи;
— Збір інформації про трафік;
— Розробка алгоритмів для пошуку розумних світлофорів на маршруті та передбачення завантаженості світлофора в момент часу;
— Фільтрація світлофорів з піковим навантаженням та побудова нового розкладу на години пік;
— Побудова маршруту, проставлення міток світлофорів, довжини та середньої тривалості поїздки, відображення графіків світлофора;
— Побудова математичних графіків ефективності алгоритму.

Web-сервіс розроблено на мовах програмування PHP та JavaScript із застосуванням технологій фреймворку Laravel, Docker, Composer, MySql, phpMyAdmin, jQuery у редакторі програмного коду JetBrains PhpStorm.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

В додатку подано опис основних структурних компонентів web-додатку для керування транспортними потоками з метою оптимізації ситуації з заторами на дорогах і виконує завдання, зазначені в розділі 1. Програмний код компонентів викладено у додатку Б.

Для використання web-сервісу необхідно мати браузер Google Chrome, Firefox, Safari або Microsoft Edge з підтримкою технології WebGL. Вимоги до операційних систем:

— Mac OS 10.12.0 або більш пізня версія;
— Windows 7 і більш пізні версії;
— Chrome OS (на комп’ютері з процесором Intel);
— Linux.

В ОС Windows XP та Vista мапа працюватиме без доступу в режим 3D і "Земля". Для роботи сервісу необхідне стабільне інтернет-з’єднання.

Web-сервіс розроблено на мовах програмування PHP та JavaScript із застосуванням технологій фреймворку Laravel, Docker, Composer, MySql, phpMyAdmin, jQuery у редакторі програмного коду JetBrains PhpStorm.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений web-додаток надає користувачу доступ до наступного функціоналу:

- Реєстрація та авторизація в системі через логін і пароль або Google OAuth2, відновлення паролю;
- Видача даних про геолокацію, затори на дорогах, дані що формують модель трафіку;
- Побудова маршруту;
- Запис поїздки та аналіз ефективності алгоритму управління транспортними потоками;
- Оптимізація графіків переключень світлофорів;
- Збереження та перегляд даних поїздки;
- Перегляд старих та оптимізованих графіків світлофорів, що лежать на маршруті.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Згідно з архітектурою, ядро програми складають алгоритми прокладання маршруту, пошуку світлофора на маршруті, передбачення завантаженості світлофора в момент часу, побудова нового розкладу на піковий проміжок.

Алгоритм прокладання маршруту реалізовано з використанням Google Maps API та TomTom Navigation API. На перший сервіс надсилається запит на пошук найоптимальнішого маршруту за часом, за довжиною, із врахуванням трафіку, а на другий — про найоптимальніших серед найчастіше використовуваних маршрутів. З-поміж отриманої колекції обирається найоптимальніший.

Алгоритм пошуку світлофора на маршруті ґрунтується на геолокації конкретного світлофора із бази даних та TomTom Search API бібліотеці, яка прогнозує ймовірність того, що точка (вузол) лежить на ребрі графа навігації.

Алгоритм побудови нового розкладу базується на методологіях GLOSA та методиці, яка є вдосконаленням роботи Ф. Вебстера, оцінкою ефективності якої є показник можливості проїзду ділянки дороги транспортним засобом в межах одного циклу світлофора, який став у колону очікування на попередньому циклі.

Логічна структура web-системи підпорядковується MVC шаблону і містить наступні логічні складові: модель, репозиторії, представлення, контролери, та сервіси.

Моделями є таблиці з бази даних системи. Репозиторії — класи, які містять методи для оперування масивами моделей. Представлення — класи, що відображають дані та структурні елементи web-сторінок (тексти, кнопки, меню, мапа, графіки і т. д.). Контролери приймають запити з frontend частини web-додатку, а в сервісах містяться методи розрахунків алгоритмів та реалізовано систему запитів до сторонніх транспортних API.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Web-сервіс розроблено на мовах програмування PHP та JavaScript із застосуванням технологій фреймворку Laravel, Docker, Composer, MySql, phpMyAdmin, jQuery у редакторі програмного коду JetBrains PhpStorm.

Для використання web-додатку користувачу достатньо мати web-браузер, вимоги до якого викладено у загальних відомостях та безперебійний доступ до мережі Інтернет.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідна інформація для побудови маршруту:

— Пункт відправки;

— Пункт призначення.

Вхідні дані для алгоритму оптимізації графіку світлофора:

— Місце розташування регульованого світлофора;

— Середня швидкість руху по маршруту;

— Якість та матеріал дорожнього полотна;

— Кількість та ширина смуг на дорозі;

— Об’єм трафіку повз світлофор за останню годину.

Вхідні дані для графіка ефективності алгоритму управління транспортними потоками:

— Довжина маршруту;

— Середня швидкість руху транспорту без заторів і з урахуванням транспортних заторів на маршруті;

— Геолокація світлофорів на маршруті;

— Активний графік переключення світлофорів;

— Якість та матеріал дорожнього полотна.

Вихідна інформація web-системи на розумний світлофор:

— JSON з оптимізованим графіком переключень світлофора.

Вихідні дані для користувача:

— Маршрут поїздки;

— Оцінка по 10-бальній шкалі заторів;

— Транспортні затори в околицях;

— Кількість керованих розумних світлофорів в місті;

— Мітки розумних світлофорів на маршруті;

— Відстань та середній час подолання маршруту;
— Старий та оптимізований графіки світлофорів на шляху;
— Математичний графік для візуалізації ефективності алгоритму управління транспортними потоками.

ДОДАТОК Г

Інструментальні засоби управління транспортними потоками

Тези на конференцію “Сучасні проблеми наукового забезпечення енергетики”

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ6168_20Б

Аркушів 3

Київ — 2020

Транспортні затори - це комплексна проблема урбаністичного, економічного та екологічного характеру, що призводить до сталих перевантажень транзитних систем перевезень і зумовлена широким спектром чинників, найбільш поширеними серед яких прийнято вважати інженерні помилки при розробці та будівництві міської інфраструктури, збільшення населення мегаполісів та швидкі темпи розвитку економіки. Наслідками цих чинників стає надмірна кількість індивідуальних транспортних засобів, значно більша за пропускну спроможність міських доріг, обмеження руху транспорту через проведення вимушених будівельних чи ремонтних робіт та автомобільні аварії, що часто блокують дорожні артерії в пікові години транспортного руху на невизначені терміни. Згідно з дослідженням Техаського транспортного інституту “2019 Urban mobility report” [1], в 2017 році американці витратили 8,8 мільярдів годин та близько 3,3 мільярдів галонів відпрацьованого топлива на затримки під час транспортного руху. За цей час 124 мільйони сімейних пар могли переглянути всі 8 сезонів серіалу Гра престолів, а показники витрат топлива відповідають лінії з 18-колісних бензовозів протяжністю від Лоса-Анджелеса до Бостона. В грошовому еквіваленті нераціональне використання робочого часу та спаленого палива для середньостатистичного американця обійшлося в $1,010.

За тривалий час боротьби із транспортними заторами урбаністи виробили ґрунтовну стратегію, що включає в себе нерозривний комплекс ефективних рішень. Нарівні з покращенням транспортної інфраструктури (розробкою оптимальних автобусних та залізничних маршрутів, побудовою метро та велосипедних доріжок), перенесенням ремонтних, будівельних, комунальних робіт на нічний час, усуненням або мінімізацією руху вантажних автомобілів міськими дорогами та багатьма іншими рішеннями виступають також грамотно розроблені системи дорожніх вказівників[2].

Поява та стрімкий розвиток інформаційних технологій дозволяють зробити цю систему динамічною і оперативно реагувати на зміни характеру та обсягів транспортних потоків.

Програмний продукт, робота якого базується на цих метриках, зібраних за допомогою мобільних пристроїв або інших обчислювальних девайсів учасників дорожнього руху дозволяє керувати світлофорами, скорочуючи або зменшуючи тривалість їх сигналів, надавати водіям прогнози щодо їх роботи, що може стати основою для системи сповіщень водіїв про затори і вибір більш оптимального шляху. Це дасть змогу більш рівномірно розподіляти транспортні потоки, економити час та пальне і навіть рятувати людські життя, уникаючи виникнення ДТП. Для отримання необхідних параметрів достатньо наявності мобільного

пристрою та стабільного інтернет-з’єднання задля забезпечення безперебійного збору валідних даних.

Для реалізації програмного продукту було обрано стек веб-технологій. Сюди входять мова розмітки HTML5, формальна мова опису стилів CSS3 та скриптова мова JavaScript з частковим використанням бібліотеки jQuery для створення інтерфейсу. Серверну частину програмного продукту реалізовано на мові PHP версії 7 та найпопулярнішому для цієї мови програмування фреймворку Laravel, 6 версії. Зібрані метрики зберігаються в реляційній базі даних MySQL. З урахуванням особливостей мови програмування, на якій було вирішено розробляти програмний продукт, в якості СКБД для розробки та подальшого управління було обрано phpMyAdmin. При розробці також було використано відкриту платформу для розробки, доставки та експлуатації розробки застосунків Docker.

Перелік посилань:

1. 2019 URBAN MOBILITY REPORT [Електронний ресурс] // The Texas A&M Transportation Institute with cooperation from INRIX. – 50. – Режим доступу до ресурсу: https://static.tti.tamu.edu/tti.tamu.edu/documents/mobility-report-2019.pdf

2. Traffic Congestion – Causes and Solution: A Study of Kota City [Електронний ресурс] // International Journal of Trend in Scientific Research and Development. – 253. – Режим доступу до ресурсу: https://www.academia.edu/36061974/Traffic_Congestion_-_Causes_and_Solution_A_Study_of_Kota_City